eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Trabajo de Fin de Grado

Grado en Ingeniería Informática

Computación

Echoes of GaIA

Framework de modelado evolutivo basado en simulación de biomas artificiales enfocado en sistemas multiagente e Inteligencia Artificial

Aingeru García Blas

Dirección

Iñaki Inza Cano

Departamento de Ciencias de la Computación e Inteligencia Artificial

Esker onak / Agradecimientos

Iñaki Inza, por tu siempre-amable personalidad. No es difícil ver por qué todo el mundo en la facultad te aprecia y tiene un cariño inconmensurable. Gracias por tus consejos y guía.

GeoAI, por haber confiado en mi y haberme dado total libertad para crear, diseñar y desarrollar el TFG totalmente por mi cuenta; lo tengo muy presente y entiendo que no todo el mundo lo hubiese hecho. Espero de verdad que os agrade.

A las siguientes profesoras y profesores por su ayuda, amabilidad y disponibilidad pese a lo pesado que soy, gracias a vosotras y vosotros este viaje ha sido maravilloso para mi. Me gustaría listar y fuertemente agradecer a tod@s a l@s que, en mayor o menor medida, os he molestado con mis inquietudes y curiosidad. Por orden alfabético:

Itziar Aldabe, Gonzalo Álvarez, Javier Alvez, Carlos Amuchástegui, Patxi Angulo, Rosa Arruabarrena, Arkaitz Artetxe, Gorka Azkune, Nagore Barrena, Mikel Barrenetxea, Idoia Bergés, Begoña Ferrero, Ibai Gurrutxaga, Montse Hermo, Mamen Hernández, Iñaki Inza, Ekaitz Jauregi, Mikel Larrea, Oier Lopez de Lacalle, Begoña Losada, Joseba Makazaga, Iñigo Mendialdua, Usue Mori, Iñaki Morlan, Maite Oronoz, Jose Antonio Pascual, Juanan Pereira, Iñigo Perona, Aitor Soroa, Julian Zapiain.

A todas y todos; habéis sido geniales - no se me va a olvidar nunca. Infinitas gracias por todo.

A mis compañeros de carrera, los que me han demostrado que, cuando hablamos de amistad, la brecha de edad es un constructo que se puede ignorar, y se pueden hacer grandes amigos allí donde no se esperaba. Un poco más, y nos vemos en el WoW cuando terminés la Universidad, venga.

A Jon Ander Peñalba, más de dos décadas diciendo tonterías juntos; y todo sigue igual, gracias por haber confiado en mi y en mis capacidades incluso cuando ni yo mismo lo hacía.

A mi padre, no sólo he heredado tu humor, si no las ganas por aprender y curiosear. No hay una sola vez que recuerde que no hayas apoyado mis decisiones. Por ser tan genial como eres y hacer que te admire, por el libro de BASIC que ha estado en la estantería desde que yo era txiki y por despertar en mí la curiosidad de cacharrear con MS-DOS en su día, mil gracias.

A Natalia, mi pareja y compañera de vida - eres única. Gracias por estar siempre ahí para todo durante tantísimos años y por tu apoyo incondicional incluso en mi decisión de - pese al tiempo que he tenido que invertir - después de tantos años, quitarme la gran espina de sacarme la carrera.

Ha sido un enorme esfuerzo hacer este viaje a estas alturas, pero ha sido mucho más fácil y especial gracias a tod@s y cada un@ de vosotr@s. Eskerrik asko bihotz-bihotzez denoi, benetan.

Laburpena / Resumen

En este proyecto presento **Echoes of GaIA** - un *framework* híbrido de simulación evolutiva que he desarrollado desde cero para modelar biomas artificiales y sus ecosistemas, tal que sean guiados por un **sistema multiagente** en el que integro diversas técnicas de *Inteligencia Artificial*. Mi idea era crear un **laboratorio virtual de ecología computacional**, con múltiples paradigmas de IA que interactúan entre sí, y con ello estudiar evolución, dinámicas climáticas/poblacionales, e incluso *comportamientos emergentes* en flora y fauna - pero, sin depender del tiempo que exige la naturaleza.

El núcleo de la simulación son **cuatro agentes especializados** que disponen de autonomía, pero que a su vez colaboran de manera pasiva: un *agente evolutivo* basado en `algoritmos genéticos` que se encarga de la selección natural y adaptación de especies; agentes de **Reinforcement Learning** que controlan el **clima** (`PPO/DQN`) y **comportamientos de fauna** (`DQN` con extractor `CNN` personalizado) para dinámicas presa-depredador; y un *agente neurosimbólico* para equilibrio ecosistémico. En este último combino de forma ponderada predicciones de **redes neuronales** con **sistemas expertos** (basados en reglas) y **análisis de grafos**.

He modelado matemática y tímidamente procesos de organismos como flora y fauna - desde *fotosíntesis*, *respiración*, *adaptación climática*... hasta *crecimiento*. También he incluido diferentes tipos de **fenómenos biológicos** e incluso los *ciclos de retroalimentación* entre **flora-fauna-clima**, junto con métricas basadas en contribuyentes para evaluar el estado general del bioma. Los mapas se representan como `grids 2D` y se generan proceduralmente con el algoritmo de **Perlin Noise**, esto es para que los entornos sean únicos en cada simulación.

Como se verá en los experimentos, especies que están **bien adaptadas** prosperan en sus biomas nativos - las que no lo están tanto (especies *desplazadas*) sufren *estrés térmico* y *déficit energético*. Pero, y he aquí donde el sistema empieza a mostrar potencial; si lanzamos simulaciones y dejamos hacer su trabajo al *agente evolutivo*, podemos observar cómo con el tiempo una especie se va integrando poco a poco en un bioma diferente del suyo; esto es, sus **genes se van adaptando**. También veremos que el *agente de Equilibrium* consigue **estabilizar simulaciones** (puntuaciones de `8.6/10` con vs `5.1/10` sin), y con ello evita desequilibrios tróficos e incluso extinciones en cascada.

He conseguido también que el agente de **Reinforcement Learning** aprenda dinámicas de *depredador-presa* con técnicas artificiales *forzadas*, como un sistema de **activación de comportamientos** en las cercanías durante el entrenamiento - que depende de su rol y estado fisiológico. Con esto y otras elementos que se expondrán, consigo que emerjan *patrones y comportamientos muy interesantes*.

Considero que resulta en un proyecto muy bonito; además, con el formato de *framework* híbrido tenemos una base modular que puede ser extendida y - a falta de mejoras - creo puede servir para estudios de *ecología computacional*. Pero hemos de tener en cuenta que modelar la naturaleza es algo **infinitamente complejo**, es por eso que he adoptado una vertiente simplista y selectiva en contraste. He tratado la naturaleza como un **sistema holístico aproximado** - la idea es que disponga de funcionalidad suficiente para que a alto nivel pueda reproducir las mismas reacciones, efectos, consecuencias...etc. y, con ello, crear un laboratorio **para hacer preguntas**; probar hipótesis sobre *cambio climático*, *evolución de flora y fauna*, *cambios de biodiversidad*... pero sin la huella que suelen dejar experimentos de campo y además, comprimir **décadas de evolución en horas de simulación**.

ODS - Aportación y reflexión

He intentado desarrollar este proyecto, a parte de por la pasión que me despierta, para intentar aportar mi pequeño granito de arena y, creo conecta principalmente con el **ODS 15** (Vida de ecosistemas terrestres) y con el ODS 13 (Acción por el clima).

Menciono el ODS 15, ya que Echoes of GaIA se trata de un *laboratorio virtual de simulaciones* para estudiar precisamente dinámicas ecosistémicas en diferentes tipos de biomas y, al ser simulación computacional - **sin generar huella experimental en campo**. Es posible analizar interacciones depredador-presa, ciclos de retroalimentación entre flora-fauna-clima, procesos biológicos y evolutivos, entre muchas otras cosas. Con ello se podría reducir la necesidad de experimentación invasiva en ecosistemas reales para ciertos estudios.

En cuanto al ODS 13, con el **agente climático** y los **modelos de adaptación térmica** que he desarrollado, podemos simular escenarios de calentamiento global y con ello analizar efectos en diferentes biomas; o incluso plantear escenarios donde un desequilibrio en la cadena trófica nos lleva a un ecosistema insostenible con altísimos niveles de CO_2; como se verá precisamente a lo largo de la memoria.

En nuestro planeta, los ecosistemas terrestres son la base de toda vida - su degradación actual es desproporcionada; en mi opinión, creo que necesitamos proteger la biodiversidad que aún nos queda, ya que **cada especie perdida es irreversible**. También me gustaria agregar que, el cambio climático - **no es un problema futuro, está ocurriendo ahora** y empeora cada año - necesitamos herramientas que nos permitan entender y actuar más rápido y, lo más complicado: concienciar a la gente.

Mi enfoque ha sido crear una **base extensible** que pueda evolucionar - es una *plataforma para explorar preguntas ecológicas complejas* y a la que otras y otros desarrollador@s puedan incorporar funcionalidades. Al igual que existen herramientas muy potentes de simulación computacional para comprender lo que ocurre en nuestro sistema solar e incluso fuera de nuestra galaxia, considero que la simulación ecológica puede acelerar nuestra comprensión, y más sabiendo que el tiempo se agota

Me gustaría que éste proyecto fuera un pequeño gesto simbólico en ésta dirección.

Índice de contenidos

CAPÍTULO 1

Introducción

1.1. Motivación personal y contexto

Para mí, este proyecto representa la culminación de mi paso por la universidad, con lo que le he dedicado mucho esfuerzo, ganas y cariño. Es un momento especial, el por fin estar en este punto después de tantos años esperando para poder sacarme la carrera, ha hecho que vea el proyecto como un summum final - pido disculpas a quien lo tenga que revisar, si resulta demasiado extenso y toca demasiadas áreas distintas. Soy consciente de la magnitud - lo he hecho por mí, ya que es un poco personal al haber tenido la espina de la carrera clavada tanto tiempo. Al haber podido disponer de libertad de acción y autonomía total para crear el proyecto - de princpio a fin - he querido poner en práctica multitud de técnicas y teorías que he ido aprendiendo estos últimos 4 años - este ha sido mi enfoque principal; basándome en ello, abrirme a investigar publicaciones científicas que antes no habría comprendido, aplicar técnicas, algoritmos e incluso modelar matemáticamente nociones que antes no se me habrían ocurrido... en general, hacer uso de mi *nueva intuición*.

Pese a haberme dedicado al desarrollo de software durante muchos años; gracias a la universidad, creo que ha cambiado mi manera de enfocar los problemas - ha ampliado y mejorado mi visión y forma de pensar. Me he volcado mucho en los estudios, y he querido centrarme en comprender y potenciar la parte matemática y resolución de problemas de forma más estructurada; he intentado transmitir precisamente esto en Echoes of GaIA, aprovechar lo que la carrera me ha dado para forzarme a salir de zonas de comfort donde antes hubiera quizá implementado una vertiente más básica o rudimentaria.

1.2. Motivación del proyecto

Cuando me paré a pensar en qué tipo de proyecto quería que mi TFG fuera, dos vertientes en cuanto a motivación fueron las que guiaron la idea. Por un lado, aprender más sobre procesos ecológicos y sus compejidades y ponerlo en práctica en su forma computacional. Por otro, la vertiente de crear desde cero un framework extensible y escalable que haga de base y disponga de un entorno tal que cualquiera pueda crear escenarios con facilidad, y utilizarlos para probar diferentes teorías ecológicas o simular evolución, ya que probar diferentes ideas que en un entorno natural real llevaría demasiado tiempo. Considero que es un proyecto interesante y que asienta las bases para una herramienta con potencial de ser extendida en un futuro y realizar simulaciones complejas que permitan hacer multitud de análisis - con lo que contribuiría a minimizar la huella experimental en campo.

Por otro lado, es un entorno bastante adecuado para explorar sinergias, he intentado llevar a cabo una especie de dinámica de sistemas. Esto es, me ha permitido poder dar un enfoque holístico e integrar técnicas que normalmente van por separado - he ahí gran parte mi motivación, a parte de lo apasionante y bonito que me parece el proyecto, es de gran complejidad.

Además, considero que podría servir también en programas educativos orientados para poder visualizar y poner en práctica conceptos complejos de ecología, biología y fisiológia con la idea de ayudar a mentalizar sobre el equilibrio natural, interacciones entre especies e incluso impacto de cambios medioambientales.

No obstante, me gustaría hacer hincapié en que idee este proyecto desde el principio para que fuera **las bases** de la herramienta que describo, para proporcionar el soporte necesario en cuanto a arquitectura de software, con la integración de módulos dedicados a modelos computacionales e inteligencia artificial.

1.3. Proyectos relacionados

Existen varios precedentes en el campo de la vida artificial y simulación de ecosistemas. Por lo que he podido ver, de los más conocidos son **Tierra** de (*Ray, 1991*[1]), el que se centra en que programas compitan por tiempo de CPU y espacio de memoria como organismos evolucionables, y **Avida** de (*Ofria y Wilke, 2004*[2]); este es una plataforma de evolución digital, que replica programas y compiten en una red de celdas. También he descubierto alguno más reciente como **ALIEN** (*ALIEN Project, 2024*[3]) que utiliza un motor de física especializado en CUDA para simular organismos digitales, e incluso **3Worlds** (*Gignoux et al., 2022*[4]), que representa ecosistemas como grafos dinámicos para facilitar comparaciones entre modelos. O incluso Eco-Simulator (*Layman, 2019*[5]), que se basa en redes neuronales para competencia por recursos de flora y fauna.

Aunque estos proyectos son muy interesantes y han establecido las bases de la simulación ecológica digital, creo que Echoes of GaIA se diferencia en varias ideas. Tierra y Avida se centran en evolución digital, y 3Worlds en representación de ecosistemas. Eco-simulator, creo abarca poco y utiliza sistemas más básicos, con lo que la utilidad es bastante limitada. Creo que en mi proyecto adopto una vertiente más holística - intento integrar múltiples paradigmas de IA en un sistema multiagente - como he comentado, combino algoritmos genéticos para evolución, reinforcement learning para comportamientos adaptativos, y sistemas neurosimbólicos para equilibrio ecológico. Además, he intentado modelar matemáticamente y paso a paso componentes biológicos como fotosíntesis, adaptación climática, nutrición, etc.

1.4. ¿Qué es Echoes of GaIA?

Al ser un framework hecho `from scratch` y, al tener un gran peso en cuanto a conceptualización, arquitectura y desarrollo (ya que son ≈18.000 lineas de código), con objeto de aliviar de carga y extensión la memoria y orientarla más a los modelos computacionales e IA, he preparado un documento anexo donde explico qué es y qué hace Echoes of GaIA, conceptos básicos, su arquitectura, Tech Stack y los sistemas principales. Esto es importante ya que aludiré a estos sistemas durante la memoria; desde el flujo de simulación, hasta el sistema de componentes, por ejemplo. Se puede encontrar en la web de Echoes of GaIA (tiene sonido), que contiene todos los links, incluido el repositorio del proyecto.

Durante la memoria se podrán ver imágenes que he utilizado para ilustrar diferentes partes, y estas pueden resultar un tanto pequeñas. Si se amplia el documento, la imagen se ampliará con buena resolución y se podrá ver sin problemas. No obstante, **también he puesto enlaces a la mayoría de las imagenes, tal que si se les hace click, llevará a la imagen original, la cual se puede ver mejor**.

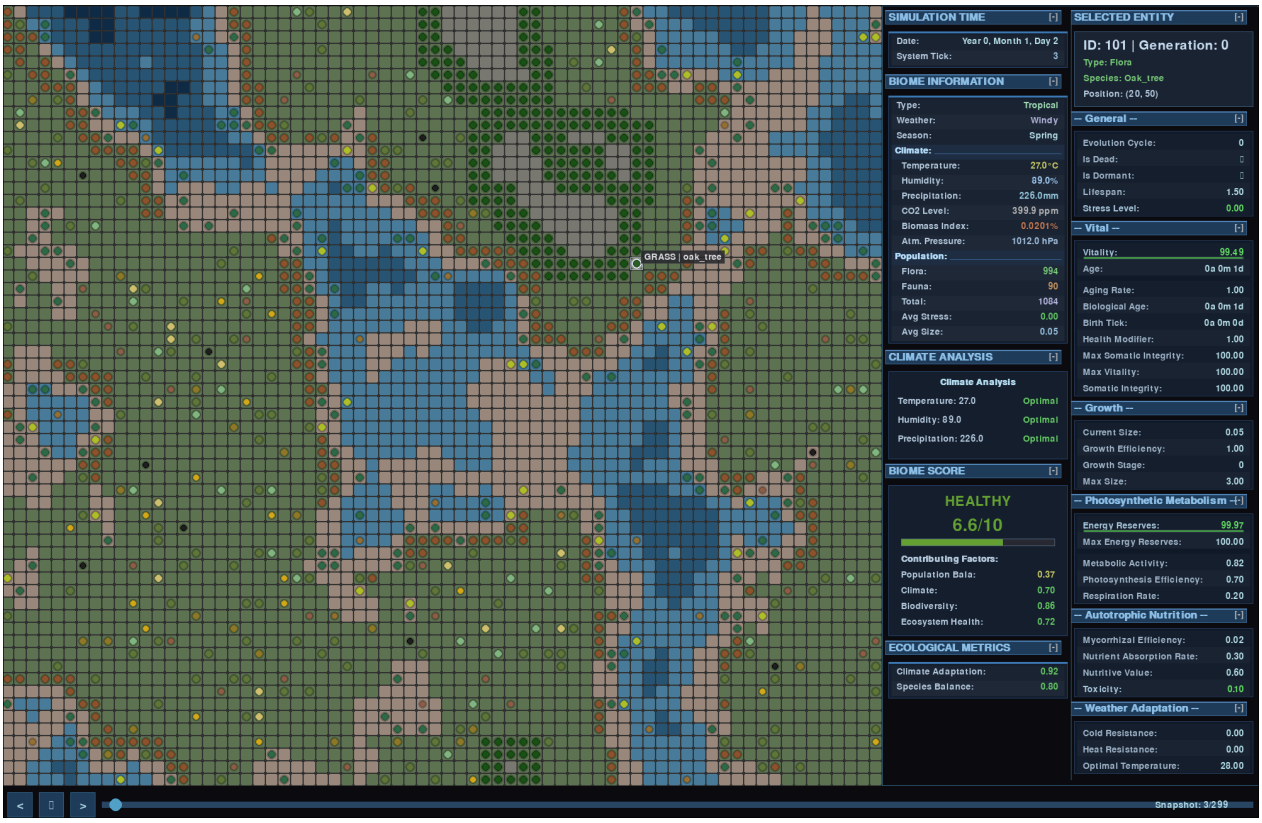


Figura 1.1: Echoes of GaIA - visor

Agradecería mucho que la lectora o el lector consultase el documento anexo antes de continuar:

Índice de contenidos del documento: Fundamentos, arquitectura y sistemas principales

Documento complementario

Ver documento de fundamentos, arquitectura y sistemas principales de Echoes of GaIA

CAPÍTULO 2

Modelo computacional de los Biomas

En este capítulo, me centro específicamente en los conceptos e ideas que permiten modelar matemáticamente la naturaleza dentro de una simulación. Así como en el aspecto computacional de algunos de los sistemas que ayudan a su representación. Tiene una gran dósis formulaica y conceptual - considero es necesaria para cuando se explique el siguiente tema, centrado en los agentes y sinergias.

El enfoque que he tomado es el de intentar diseñar un modelo computacional que me sirva para simular y estudiar el comportamiento de ecosistemas; he utilizado diferentes técnicas algorítmicas y modelos matemáticos que intentan expresar o emular la biología y física de los organismos. El objetivo al que he intentado llegar es el de automatizar virtualmente lo que la intuición, simples soluciones o cálculos analíticos no consiguen (o es muy difícil de conseguir).

Durante este viaje universitario, he cursado muchas asignaturas que me han ayudado a robustecer fundamentos matemáticos, considero que pensar en esos niveles de abstracción me han proporcionado una manera diferente de enfocar los problemas y, eso lo he notado mucho cuando he trabajado en darle forma a los diferentes organismos y sistemas del bioma. He aprovechado este proyecto, para poner en práctica esto y aplicar tanto cosas aprendidas, como continuar explorando. Allá donde antes quizá hubiera optado por realizar una implementación simplista basada en reglas, ahora me paro a pensar posibles formas más complejas, relaciones no lineales y la manera de modelar incluso en múltiples dimensiones. Por eso he disfrutado tanto de hacer lo que voy a exponer en este tema, además de haber adquirido, a un nivel modesto, la habilidad de informarme y captar ideas leyendo papers de investigación científica - he podido complementar para desarrollar un bioma que emule, muy tímidamente, la naturaleza.

Menciono tímidamente, por que obviamente modelar el comportamiento de organismos vivos y sus ecosistemas es algo extremadamente complejo. Con lo que, reconozco que el enfoque aplicado sigue siendo relativamente simple si lo comparamos a la complejidad inherente del mundo real, ademas de algo selectivo en cuanto a selección de comportamiento o cualidades (no se puede abarcar todo). Al construir un sistema totalmente desde cero, considero hay que comenzar desde modelos más sencillos e ir aumentando su complejidad gradualmente, ya que intentar empezar por la parte más complicada conduce inevitablemente a una infinidad de problemas. Además, gracias al diseño, Echoes of Gaia se puede extender y ampliar funcionalidad de manera sencilla.

Veamos el estado actual de los modelos computacionales, las decisiones que los sostienen y las vías de mejora que considero han quedado abiertas.

2.1. Mapas y gestión espacial

2.1.1. Generación procedural de mapas: algoritmo Perlin noise

Como vimos en la figura 1.11 del documento de Arquitectura y sistemas, necesitamos generar mapas con índices de terrenos que entre sí, formen patrones coherentes y no repetitivos - existen diversas formas pero desde hace mucho tiempo venía escuchando hablar de **Perlin noise** para generar mapas procedurales en videojuegos - recuerdo que antiguamente se implementaba el algoritmo o con plugins, pero hoy día los motores principales en el mercado lo han integrado y expuesto a través de sus APIs:

- Unity: Mathf.PerlinNoise | | Unreal5: PerlinNoise2D

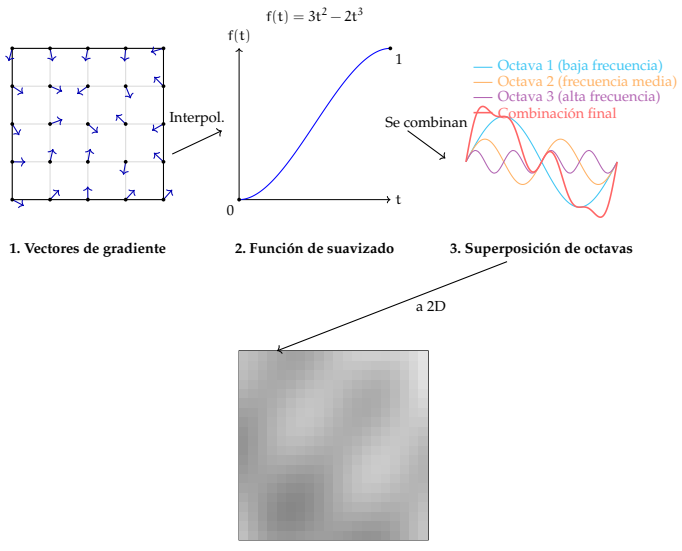
Yo utilizo el paquete de perlin-noise en el repositorio oficial de Python.

2.1.1.1. Alternativas Consideradas

Antes de decidirme por este algoritmo, investigué un poco sobre otro muy sonado también: **Wave function collapse**, pero no encontré librerías ni implementaciones de código abierto que fueran fácilmente integrables y, como la generación del mapa no es algo a lo que quisiera dedicar mucho tiempo dada la envergadura del proyecto, opté por Perlin. No obstante, lo poquito que he visto, me ha parecido un algoritmo fascinante - me lo he apuntado para un futuro.

2.1.1.2. Funcionamiento de Perlin Noise

Con espíritu investigador, aunque no fuera a implementar el algoritmo yo, he investigado un poco sobre cómo funciona internamente. El ruido Perlin funciona superponiendo funciones de ruido a diferentes escalas (octavas), lo que produce gradientes suaves que se asemejan a formaciones naturales:



1. Vectores de gradiente **2. Función de suavizado** **3. Superposición de octavas**

4. Cada punto (x,y), calculamos el valor combinado de todas las octavas

De casualidad, leyendo sobre Perlin, me topé con este paper muy interesante que me ha enseñado otro método para inicializar redes neuronales con perlin noise; sirve para obtener filtros visuales más ricos desde el principio en CNNs (se centra en ResNet) - hasta ahora únicamente había utilizado Xavier/Gorot, He/Kaiming y Orthogonal(*Inoue et al., 2021*[6])

- **1. Cómputo de los vectores gradientes**

  Para calcular los vectores se utiliza un algoritmo que asigna un vector unitario pseudoaleatorio a cada punto de un grid. Se escoge un punto, se decide la celda a la que pertenece y se toman los cuatro gradientes (uno de cada esquina).

- **2. Suavizado**

  AHora, esos cuatro gradientes, se interpolan, pero se suavizan con unos pesos. Estos pesos se calculan con la función:

$$f(t) = 3t^2 - 2t^3$$

  A la que se le pasa el punto (x,y). Se conoce que se escogió inicialmente esta función porque interpola entre 0 y 1 tal que $f(0) = 0$, $f(1) = 1$ y las derivadas en 0 y 1 valen 0 (transición suave). Pero, en el paper original de (*Perlin, 1985*[7]), no la menciona explícitamente, solo comenta que se computa una interpolación suave, y da como ejemplo un polinomio cúbico (sin especificar) pero, es la función que está extendida en la mayoría de los ejemplos que se ven. También mapea los valores entre 0 y 1. Posteriormente, en su paper de (*Perlin, 2002*[8]), ya sí que menciona explícitamente:

> $3t^2$ $2t^3$ is replaced by $6t$ $15t + 10t^3$, which has zero first and second derivatives at both t=0 and t=1.

- **3. Combinación de octavas**

  Ahora, se generan octavas que aplican el ruido con distinta frecuencia/amplitud – se modifican tal que se duplica la frecuencia y se reduce la amplitud:

$$\text{valor} * \text{ruido} = f(p) + 0{,}5\, f_2(p) + 0{,}25\, f_4(p) + 0{,}125\, f_8(p) \tag{2.1}$$

  Siendo $f(p)$ el valor escalar que nos ha devuelto la interpolación ponderada de los gradientes. Como nota extra, aquí, quise entender por qué se duplica la frecuencia y cómo afecta – ha resultado en que, en cada paso donde se duplica la frecuencia, es como si estuvieramos escalando el mapa de gradientes.

- **5. Implementación en código**

  Como he comentado anteriormente, no quería dedicarle mucho tiempo a ésta parte de generación procedural (aunque, al final, me ha terminado encantando y he aprendido bastante investigando sobre ello), con lo que me he inspirado en un código disponible públicamente (ver código) (sin licencia de Software Libre específica).

  Éste código utiliza 4 generadores o capas Perlin Noise con diferentes frecuencias (3, 6, 12, 24), después, para cada punto combina los valores de esos 4 generadores de la misma manera en la que uno solo combina las octavas internamente. Yo lo he adaptado al proyecto con algo como:

```
perlin_3_freq: Callable = PerlinNoise(octaves=3, seed=self._seed)
perlin_6_freq: Callable = PerlinNoise(octaves=6, seed=self._seed)
perlin_12_freq: Callable = PerlinNoise(octaves=12, seed=self._seed)
perlin_24_freq: Callable = PerlinNoise(octaves=24, seed=self._seed)
.....
noise_value = perlin_3_freq(tuple(normalized_coords[y, x]))
```

```
noise_value += 0.5 * perlin_6_freq(tuple(normalized_coords[y, x]))
noise_value += 0.25 * perlin_12_freq(tuple(normalized_coords[y, x]))
noise_value += 0.125 * perlin_24_freq(tuple(normalized_coords[y, x]))
self._map.noise_map[y, x] = noise_value
```

He mantenido misma semilla para todos, para que no sea muy caótico (si no cada banda será diferente). Lo primero que me planteé, es que esto, **aporta mucha más información y, si hay más información, habrá mayor nivel de detalle**. Pero quería ver lo que ocurría si utilizaba únicamente un generador o capa, ya que internamente ya calcula la amplitud de sus octavas. Efectivamente, tras cargar en el visor los mapas con una sola capa y bajas octavas, se aprecia que tienen patrones más extensos y suaves. Los producidos con las cuatro capas Perlin Noise, tienen más subzonas y detalles aislados – me he quedado con ésta versión.

También he probado con una sola capa y muchas más octavas – queda muy diferente, asumo que es por la sobrecarga repetida de octavas de baja frecuencia, al tener las cuatro capas. Mi conclusión es que, una capa con relativamente pocas octavas (3–12) puede ser muy bueno para generar mapas 2D con nubes, océanos etc., pero si ya queremos más nivel de detalle y zonas *localizadas* – creo que la idea que tomé del código de CodingQuest es muy buena.

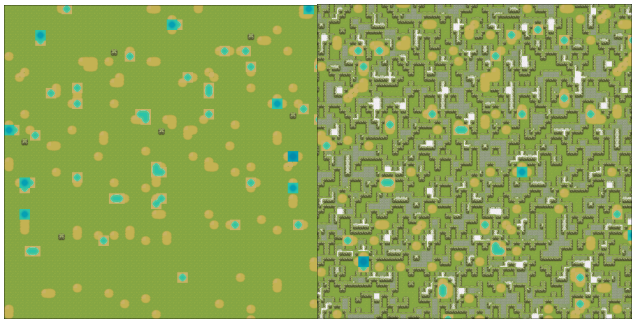


Figura 2.1: Perlin 1 capa 24 octavas || 1 capa 24 octavas

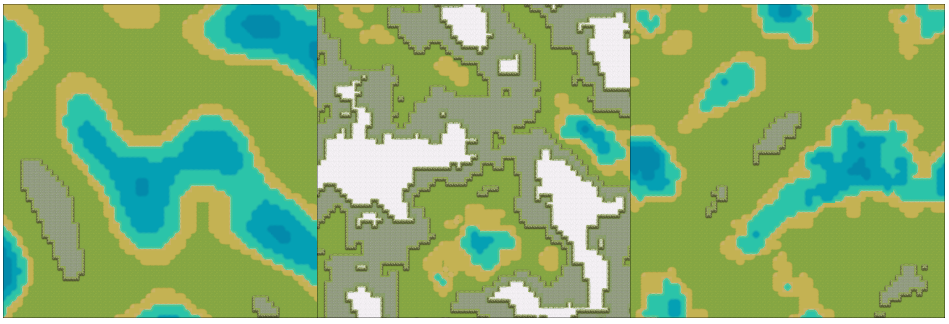


Figura 2.2: Perlin 1 capa 3 octavas || Perlin 4 capas 24 octavas || Perlin 4 capas 24 octavas

Viendo esto, sabemos que este algoritmo es tan utilizado porque produce un ruido muy suavizado de manera que las transiciones entre un punto y otro son sutiles y no parece **ruido puro**; es una *randomización coherente*. También – **escalabilidad**; esto es extremadamente útil, ya que no importa si lo utilizo en un mapa 10x10 que en uno 100x100, el patrón será generado consistentemente

Ahora, nos falta transformar el noise map o mapa de ruido Perlin, a nuestro terreno.

2.1.1.3. Pesos para los biomas

Necesito asignar pesos dependiendo de la distribución de terrenos que desee, y esto a su vez depende del tipo de bioma. Con lo que, habiendo asignado el orden de los pesos tal que:

`[WATER_DEEP, WATER_MID, WATER_SHALLOW, SHORE, GRASS, MOUNTAIN, SNOW, SAND]`

Estas son las distribuciones de pesos por bioma que tengo en **ecosystem.json**:

- **Tropical:** [11, 12, 13, 11, 50, 0, 0] - Predominancia de vegetación con áreas de agua
- **Desert:** [0, 5, 5, 10, 10, 3, 0, 50] - Zonas muy extensas de arena, vegetación escasa
- **Taiga:** [5, 10, 15, 10, 30, 25, 5] - Bosques y terreno montañoso con nieve
- **Savanna:** [0, 5, 10, 10, 70, 5, 0] - Llanuras con un poco de agua y alguna montaña
- **Tundra:** [0, 5, 10, 10, 20, 20, 0] - Áreas nevadas

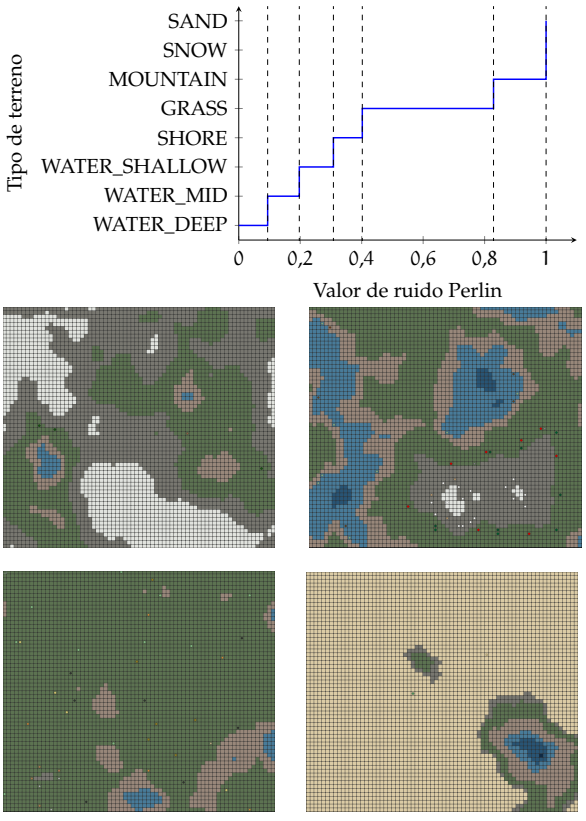
2.1.1.4. Transformación del ruido en terreno

Ahora, vamos a convertir ese mapa de ruido en nuestros tipos de terrenos. El proceso se puede ver aquí: ver código. Esto se hace con umbrales normalizados que calculamos a partir de los pesos; se divide el rango de valores del ruido en segmentos, y estos, se asocian a cada tipo de terreno:

$$\text{terreno}(x,y) = \begin{cases} \text{WATER\_DEEP}, & \text{si ruido}(x,y) < h_1, \\ \text{WATER\_MID}, & \text{si } h_1 \le \text{ruido}(x,y) < h_2, \\ \text{WATER\_SHALLOW}, & \text{si } h_2 \le \text{ruido}(x,y) < h_3, \\ \text{SHORE}, & \text{si } h_3 \le \text{ruido}(x,y) < h_4, \\ \text{GRASS}, & \text{si } h_4 \le \text{ruido}(x,y) < h_5, \\ \text{MOUNTAIN}, & \text{si } h_5 \le \text{ruido}(x,y) < h_6, \\ \text{SNOW}, & \text{si } h_6 \le \text{ruido}(x,y) < h_7, \\ \text{SAND}, & \text{si } h_7 \le \text{ruido}(x,y). \end{cases} \tag{2.2}$$

(2.2)

En este proceso, todo es importante, pero tiene especial relevancia la función `cumsum` de **numpy**, lo hace de manera vectorizada, pero veamos un ejemplo de cómo se calcula la probabilidad acumulativa:



Cálculo de umbrales:

$r = \text{ruido}(x,y)$,
$w = [11, 12, 13, 11, 50, 20, 0]$ (pesos),
$W = 117$ (suma total),
$p = [0{,}094, 0{,}103, 0{,}111, 0{,}094, 0{,}427, 0{,}171, 0]$ (norma.).

Umbrales acumulativos:
$t = [0{,}094, 0{,}197, 0{,}308, 0{,}402, 0{,}829, 1{,}000, 1{,}000]$
Cálculo general: ($t_{\text{mín}} = 0, t_{\text{máx}} = 1$)

$$t_i = t_{min} + (t_{max} - t_{min}) \cdot \sum_{j=1}^{i} p_j \tag{2.3}$$

Asignación de terreno:

$$\text{terreno}(x,y) = \begin{cases} \text{WATER\_DEEP}, & r < 0{,}094 \\ \text{WATER\_MID}, & r < 0{,}197 \\ \text{WATER\_SHALLOW}, & r < 0{,}308 \\ \text{SHORE}, & r < 0{,}402 \\ \text{GRASS}, & r < 0{,}829 \\ \text{MOUNTAIN}, & r < 1{,}000 \\ \{\text{SNOW}, \text{SAND}\}, & r \ge 1{,}000 \end{cases} \tag{2.4}$$

Figura 2.3: Biomas tundra, taiga, sabana y desierto.

2.2. Gestión de hábitats

2.2.1. Precómputo de caché de hábitats

2.2.1.1. Definición de Hábitat

En el modelo que he diseñado, un hábitat se define mediante dos criterios básicos:

- **IN:** El terreno donde la entidad existe directamente.
- **NEARBY:** Los terrenos adyacentes que influyen en su aptitud para habitar ese lugar.

Ver definiciones de hábitats.

Al definir los hábitats tal que así, me es posible implementar un bioma donde cada entidad tiene preferencias específicas. Esto hace que su localización inicial dependa de condiciones de hábitat cuando es instanciada - esto es muy útil para la flora, ya que la fauna terminará desplazándose. Igualmente, ayuda a crear patrones de distribución espacial más ricos.

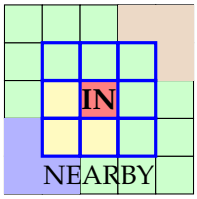


Figura 2.4: Representación del concepto de hábitat con zonas IN y NEARBY

Una vez generado el mapa de terreno, necesito determinar qué áreas son adecuadas para cada especie. Existen procesos que consultarán disponibilidad de hábitats o realizarán operaciones sobre ellos - esto puede resultar en sobrecarga considerable. Por eso, nada más empezar a pensar en este sistema, vi claro que necesitaba algún tipo de **caché con acceso de orden constante**. Durante la inicialización, analizo todo el mapa y la distribución de terrenos para crear la asociación.

Generación y asignación de hábitats

> Este punto lo implementé en una fase temprana del proyecto, y fue el que me hizo darme cuenta de cómo mi visión estaba cambiando gracias a estos años últimos 4 años en la universidad, pese a tanto tiempo en el mundo profesional. Allá donde antes hubiera ideado alguna forma simplista, ahora en seguida me vino a la cabeza la **operación de convolución** a la que nos introdujeron en la asignatura de Aprendizaje Automática y Redes Neuronales (AARN) para las CNNs; necesitamos recorrer el mapa y asignar a cada celda su hábitat según sus vecinos - si lo hacemos de forma **Naîve**, esto son 4 bucles anidados; si el mapa es grande, pues resulta muy ineficiente.

2.2.1.2. Operaciones de convolución para detección de hábitats

La idea central es identificar patrones específicos en el mapa deslizando convoluciones por todas las celdas.

$$K_{\text{borde}} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{2.5}$$

En mi implementación, expando este kernel a 4x4 con `np.pad` para aumentar el área de detección.

Para cada tipo de hábitat, aplico la operación de convolución con este kernel sobre el mapa de terreno. Esto hace que cada celda obtenga información sobre sus vecinas. Matemáticamente, lo expreso así:

$$\text{hábitat}(x, y) = \sum_{i=-1}^{1} \sum_{j=-1}^{1} K(i, j) \cdot \text{compatible}(\text{terreno}(x + i, y + j)) \tag{2.6}$$

La función *compatible* evalúa si una celda vecina $(x + i, y + j)$ coincide con algún tipo especificado en NEARBY para el hábitat actual. Esta evaluación solo se ejecuta si la celda central (x, y) pertenece a un terreno lisatdo en `IN`. Entonces, si la celda vecina coincide y el valor correspondiente en el kernel es 1: contribuye al resultado.

Una vez hechas todas las convoluciones, **si el valor en una celda es mayor que cero, implica que al menos tiene una vecina que cumple la condición `NEARBY`, y por ello, pertenece al tipo de hábitat** que buscamos.

Veamos un ejemplo visual para una celda que cumple las concidiones del hábitat `tropical_wetland`: la **(x,y) = (1,1)**:

Tropical wetland

- **IN:** `WATER_SHALLOW (2), SHORE (3)`
- **NEARBY:** `GRASS (4)`

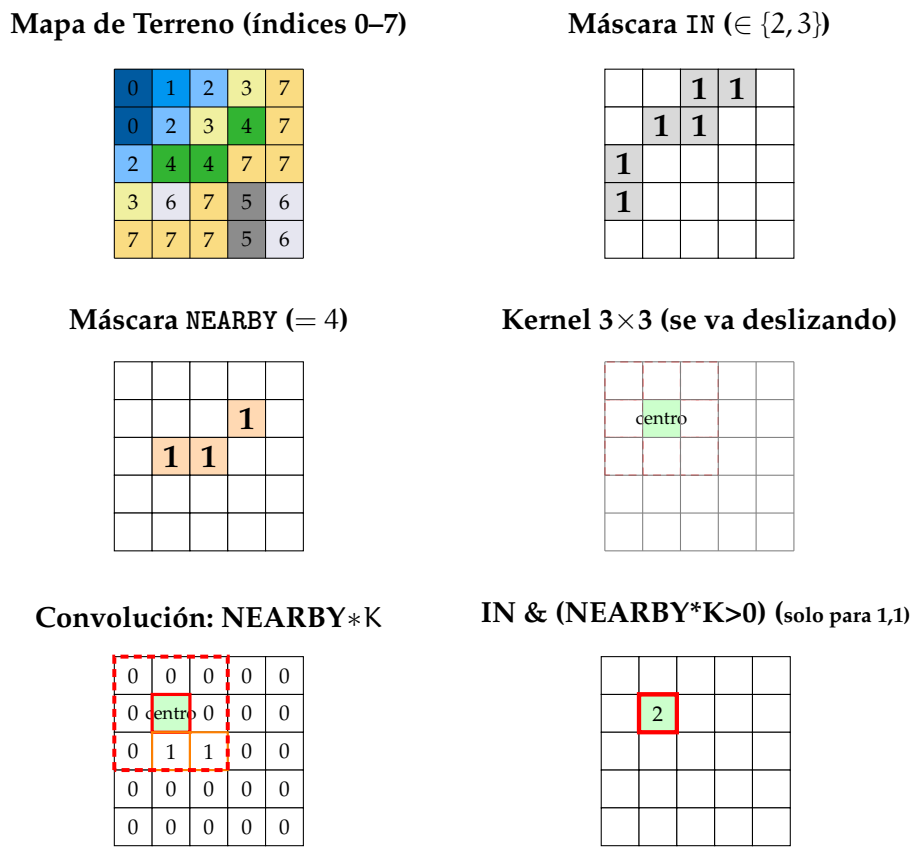


Figura 2.11: Proceso convolución con kernel 3×3 (sin expansión) para el hábitat `tropical_wetland`.

Así obtengo la máscara de hábitats para éste hábitat actual, y así puedo registrar en la **estructura de datos que hace de caché las posiciones válidas.**

Esto no me salió bien a la primera. Me despisté y no me percaté de que las máscaras binarias en `Python` con `NumPy` se comportan, efectivamente, como máscaras binarias *bit a bit*. Estoy acostumbrado a usarlas en `C++` para distintas mecánicas, pero no vi al principio que el comportamiento era exactamente el mismo.

Por ejemplo, si hago algo como `in_mask & nearby_presence`, el resultado será `1` solo si **ambos** son `1`. Si uno de los dos es diferente (por ejemplo `2` o `0`), el resultado será `0`, ya que `&` compara bit a bit y solo da `1` cuando los dos bits en la misma posición son `1`.

El problema es que `nearby_presence` proviene de una convolución, y puede contener valores **mayores que 1** si hay varias celdas vecinas activas. Para que funcione correctamente como máscara booleana, he transformado los valores

```
nearby_presence = (nearby_convolved > 0).astype(np.int8)
```

para que todo valor mayor que `0` se convierta en `1`.

2.2.1.3. Algoritmo

Ver código de proceso de convoluciones

2.2.2. Análisis de optimización

2.2.2.1. Complejidad

Enfoque	Descripción	Complejidad temporal
Naive	Cuatro bucles anidados para recorrer todo el grid 2D; analiza para cada celda sus vecinos.	$O(H \times M \times N \times k^2)$
Convoluciones	Deslizo un kernel de convolución sobre el mapa.	$O(H \times M \times N \times k^2)$

H = núm. de tipos de hábitats; $M \times N$ = tamaño del mapa (filas × columnas); k^2 = núm. de elementos del kernel.

Como se puede observar, la complejidad teórica es la misma, pero la diferencia está en la implementación en C de NumPy y SciPy, ya que disponen de optimizaciones internas que hacen que en la práctica sea muchísimo más rápido que iterar explícitamente.

2.2.2.2. Resultados

He hecho unas pruebas con ambas versiones, veamos los resultados:

Resultados	
Tamaño del mapa	50×50
Tamaño del kernel	4×4
Número de hábitats	1
Rendimiento	
Tiempo total (Naive)	32,60 ms
Tiempo total (`conv`)	0,11 ms
Aceleración	$\approx 296\times$
Accesos a memoria (Naive)	82 500

Tabla 2.1: Métricas de rendimiento para ambos enfoques

Pese a tener el mismo orden temporal y accesos a memoria, la diferencia en tiempos es enorme - la aceleración ganada por las optimizaciones internas es muy alta. Estas optimizaciones aprovechan SIMD, paralelización etc, y también se benefician del hecho de que los accesos a memoria son contiguos. He decidido contar los accesos del método Naive para escritura, para convoluciones sabemos que son los mismos. Sin embargo – la contigüidad justifica parte de esta aceleración; implica alta eficiencia de caché con técnicas como prefetch etc. - con lo que podemos asumir que reduce hasta en **2x el tiempo de acceso respecto la versión Naïve**. El propio artículo de referencia para esto (*Walt et al., 2011*[9]) indica que estas operaciones, al evitar copias y aprovechar strides y contigüidad, son extremadamente eficientes y la causa de las mejoras de rendimiento.

Nota. También podría calcularse la convolución mediante *FFT* (*Fast Fourier Transform, (Virtanen et al., 2020[10])*), **pero el coste de preparar la transformada para *kernels* pequeños no compensa**. El orden temporal de la *FFT* es

$$\mathcal{O}\left(N^2 \log N^2\right).$$

Básicamente, esto significa que, mientras k crece, $\mathcal{O}(NMk^2)$ lo hace cuadráticamente; pero con *FFT*, lo hace muchísimo más lento gracias al logaritmo. No obstante, como he utilizado 4×4, me quedé con el `convolve` simple.

2.2.2.3. Consideraciones

¿Merece la pena? Desde luego, esta optimización es positiva con vistas a que el proyecto escale. Pero hay que tener en cuenta que es un **precómputo** - sólo ocurre durante el Bootstraping (2.5.7 en el documento arquitecture) y la diferencia real es de 32.60ms a 0.11ms - diferencia importante. La idea de implementar este sistema sería más fuerte si el cálculo se hiciera con frecuencia - no obstante, considero que estas optimizaciones son correctas porque si el proyecto crece o se prueban escenarios más grandes, la diferencia será mucho más notable en la inicialización, y esto tiene **especial relevancia durante el entrenamiento del agente de fauna**, como se verá más adelante.

Donde realmente se nota cómo se guardan estos datos de habitat-posiciones es en el movimiento de la Fauna. Cuando una entidad está en una celda, esa celda contiene el valor del **ID** de la **entidad en el mapa de índices** - si se desplaza, hay que poner el valor a -1 para liberar la celda original y ocupar la nueva. Pero claro, cada una de estas celdas pertenece a un hábitat.

En cada movimiento hay que gestionar la lógica de hábitat para ambas celdas - liberar la antigua y ocupar la nueva, así cuando una entidad nace puede hacerlo en su hábitat correspondiente. La estructura de datos tiene este aspecto:

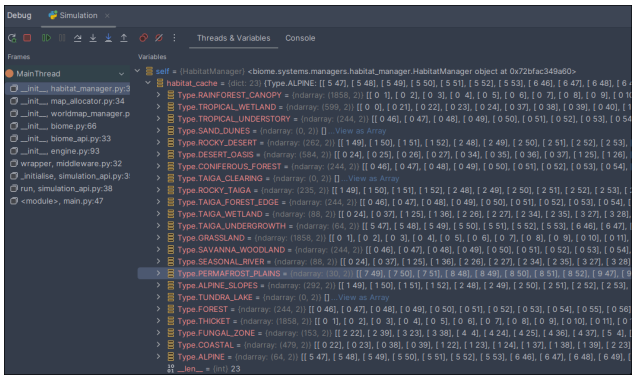


Figura 2.12: Caché de hábitat, sacada del debugger.

El acceso para la ocupación/liberación de hábitats con fauna en movimiento es de orden **O(1)**, ya que uso *np.rand.randint()* para seleccionar un índice aleatorio de posición. Para restaurar el hábitat

de una celda que se acaba de liberar, el orden es $O(P)$, donde P es el número de posiciones; para evitar duplicados se comprueba.

2.3. Modelo climático

Como he mencionado anteriormente, el primer enfoque para modelar muchos de los aspectos climáticos y naturales ha sido una vertiente simplista, lo cual he ido mejorando iterativamente y, es algo que ahora fácilmente puede seguir siendo extendido. El clima se define mediante un conjunto de variables interrelacionadas que conforman el estado climático:

Temperatura **humedad** **precipitación** **nivel de CO_2** **presión atmosférica** **densidad de biomasa** **densidad de fauna**

2.3.1. Efectos de retroalimentación

Estas variables no son independientes; forman un sistema interconectado con retroalimentación. Se irá viendo a lo largo del proyecto más en detalle, pero para dar una idea, veamos un ejemplo:

1. La temperatura afecta las tasas de transpiración de la flora
2. La transpiración afecta los niveles de humedad atmosférica
3. La respiración y fotosíntesis de la flora afectan los niveles de CO_2
4. Los niveles de CO_2 influyen en la temperatura; efecto invernadero

Todo lo que ocurra en el sistema, ya sea relacionado a la biomasa o a la fauna - afecta al clima. Y a su vez lo que ocurra en el clima afecta a la vida en el Bioma, con lo que sea crea un bucle de retroalimentación. Este fenómeno lo modelo de manera distribuida de forma que las entidades gestionan sus procesos en función al entorno y proporcionan feedback sobre su estado. A su vez, el sistema climático modela su parte de la contribución con ecuaciones como:

2.3.2. Cálculo de biomasa, densidad de fauna y factores de flora

$$\rho_{\text{biomasa}} = \frac{1}{|F|} \sum_{i \in F} \frac{\min(s_i, s_i^{\text{máx}})}{s_i^{\text{máx}}} \tag{2.7}$$

$$\Phi_{\text{total}} = \sum_{i \in F_p} \phi_i \, \alpha_i \, \frac{s_i}{s_i^{\text{máx}}} \cdot 0{,}5 \tag{2.8}$$

$$R_{\text{total}} = \sum_{i \in F_p} r_i \, \frac{s_i}{s_i^{\text{máx}}} \, \alpha_i \cdot 0{,}5 \tag{2.9}$$

$$Tr_i = \phi_i \, \alpha_i \, \min\left(1, \frac{T}{30}\right) 0{,}1 \left(1 - 0{,}7 \, \frac{\sigma_i}{\sigma_i^{\text{máx}}}\right) \frac{s_i}{s_i^{\text{máx}}} \tag{2.10}$$

$$Tr_{\text{total}} = \sum_{i \in F_p} T_i \tag{2.11}$$

$$\rho_{\text{fauna}} = \frac{1}{|A|} \sum_{j \in A} \frac{\min(s_j, s_j^{\text{máx}})}{s_j^{\text{máx}}} \tag{2.12}$$

	Flora
F	Conjunto de entidades de flora vivas.
F_p	Subconjunto con metabolismo fotosintético.
s_i	Tamaño actual de la entidad i.
$s_i^{\text{máx}}$	Tamaño máximo de la entidad i.
ϕ_i	Eficiencia fotosintética .
α_i	Actividad metabólica.
r_i	Tasa de respiración.
σ_i	Nivel de estrés.
$\sigma_i^{\text{máx}}$	Estrés máximo.
Tr	Transpiración.
	Fauna
A	Conjunto de entidades de fauna vivas.

Nota 1: Algunos de estos valores son parámetros de componentes (definidos por config) y otros son calculados - se irán exponiendo. **Nota 2:** Algunos factores constantes han sido fruto del ajuste tras experimentación, sin embargo los factores escalares de 0.5 que ponderan las ecuaciones de eficiencia fotosintética y respiración, las ajusto de manera que emule que la mitad del día la flora esté produciendo CO2 ligeramente (cuando no hay luz), y la otra mitad está realizando fotosíntesis (cuando sí hay luz).

2.3.3. Ecuaciones de balance CO_2 y efectos ambientales

$$\Phi_{CO_2}^{fauna} = \rho_{fauna} \cdot \kappa_{fauna} \quad (2.13)$$

$$\Phi_{CO_2}^{flora} = \rho_{biomasa} \cdot \kappa_{flora} \quad (2.14)$$

$$\Delta_{CO_2} = \Phi_{CO_2}^{fauna} - \Phi_{CO_2}^{flora} \quad (2.15)$$

$$CO_2(t+1) = CO_2(t) + \Delta_{CO_2} \quad (2.16)$$

$$\Delta T_{CO_2} = 0{,}01 \cdot (CO_2(t) - 400) \quad (2.17)$$

$$T(t+1) = T(t) + \Delta T_{CO_2} \quad (2.18)$$

$$\Delta H = 3 \cdot Tr_{total} \quad (2.19)$$

$$H(t+1) = H(t) + \Delta H \quad (2.20)$$

Símbolo	Descripción
$\kappa_{fauna} = 0{,}8$	Emisión de CO_2 por unidad de fauna.
$\kappa_{flora} = 7{,}0$	Absorción de CO_2 por unidad de biomasa.
$CO_2(t)$	Nivel de CO_2 en tiempo t.
ΔT_{CO_2}	Lo que varía la temperatura en función del CO_2 $T(t)$ Temperatura en tiempo t.
$H(t)$	Humedad en tiempo t.

Nota: he usado ρ para densidad, ϕ para flujos, α para coeficientes, τ para rates y otras griegas según me ha sido posible.

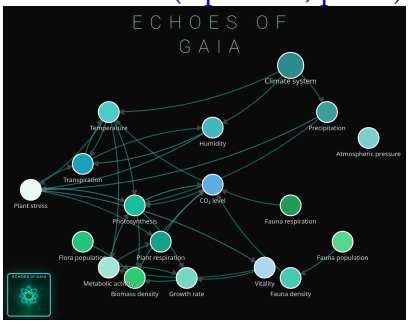
Bien, en este modelo he intentado extraer relaciones funcionales básicas de principios ecológicos. He representado factores como fotosíntesis, respiración, transpiración y emisión de CO_2 mediante funciones simplificadas pero **inspiradas** en modelos reales. Por ejemplo, la respiración vegetal aumenta con la temperatura siguiendo una relación tipo Q10 (*Atkin y Tjoelker, 2003*[11]) (se menciona que x2 cada 10 grados), mientras la transpiración la he modulado según temperatura y estrés (*Allen et al., 1998*[12]).

He considerado que el CO_2 atmosférico afecta a la temperatura de forma general (*Intergovernmental Panel on Climate Change (IPCC), 2013*[13]) - mencionan que el aumento de CO_2 contribuye al calentamiento global, con efectos en cascada sobre procesos fisiológicos como fotosíntesis y respiración. Yo, de aquí interpreto que la relación no es lineal y su efecto no se produce a tiempo real, así que he hecho una simplificación con relación efecto-causa directa para adaptarlo a la simulación; los factores pueden ajustarse para ver el efecto a largo plazo. También me he basado en la noción de que el balance de carbono depende tanto de la absorción por biomasa como de la emisión por fauna (*NASA, 2011*[14]).

Las ecuaciones no replican modelos originales - son muy complejos para el tiempo de desarrollo del TFG - pero he intentado condensar nociones documentadas en la literatura, y las he ajustado aplicando **(mi) lógica y sentido común - y mucha experimentación empírica en el simulador**; principalmente para los factores escalares. La fotosíntesis, por ejemplo, se calcula a partir de eficiencia, metabolismo y tamaño (*Farquhar et al., 1980*[15]) y se reduce por estrés ambiental (*Flexas et al., 2004*[16]). La densidad de fauna y capacidad de absorción vegetal están acotadas por factores y límites con la intención de ser realistas, inspirados en (*Chapin III et al., 2011*[17]), donde implícitamente usan capacidades máximas - por esto se podrá apreciar que he acotado explícitamente (en el código, he obviado en muchas ecuaciones por limpieza, aunque en algunas he considerado aportaban información dejarlo (`min/max`).

Como expliqué en el documento de fundamentos y arquitectura, los componentes gestionan la lógica biológica de las entidades. Considero que son gran parte del alma de la simulación y he dedicado bastante tiempo a construirlos, investigando y probando. Me gustaría exponer cómo los he modelado finalmente. Se puede consultar **Componentes y sus atributos** (Apéndice, p. 114)

Utilizaré el término *xxx efectiva* (fotosíntesis efectiva, tasa de absorción efectiva...); son valores finales que determinan ganancias/pérdidas en cada instante. En general casi siempre habrá valores base y, unos valores dinámicos que irán cambiando durante la simulación – lo bonito es que estos valores base son **genes** de la especie, que irán mejorando conforme avance la simulación. Más sobre esto en el apartado de **evolución**. Ver grafo interactivo



2.4. Componentes biológicos

2.4.1. Componente vital

Principalmente sirve para gestionar los **parámetros de salud y envejecimiento.** Se centra en el cálculo y actualización de la vitalidad, integridad somática, estrés y envejecimiento de las entidades del ecosistema. Este componente dicta cuándo la entidad muere y gestiona el deterioro natural a lo largo de su vida.

Figura 2.13

2.4.1.1. Cálculo de envejecimiento

Se calcula la edad biológica de una entidad en función del tiempo de simulación y un modificador de tasa de envejecimiento:

$$A_{bio} = A_{chron} \cdot r_{age} \tag{2.21}$$

Símbolo	Descripción
A_{bio}	Edad biológica.
A_{chron}	Edad cronológica.
r_{age}	Tasa de envejecimiento.

La tasa de envejecimiento puede variar según factores genéticos y ambientales; algunas entidades envejecerán más rápido o más lento que otras; puede depender de si más o menos estrés sufrido...por ejemplo. La edad cronológica se basa directamente tiempo de simulación:

$$A_{chron} = \frac{T - T_0}{\tau} \tag{2.22}$$

Símbolo	Descripción
A_{chron}	Edad cronológica.
T	Tick actual.
T_0	Tick de nacimiento.
τ	Número de ticks por año.

2.4.1.2. Deterioro de vitalidad: modelo de Gompertz

En un principio comencé con una curva sigmoidea inversa para modelar cómo se va reduciendo la vitalidad. No me terminaba de convencer, porque en el centro de la vida, la pendiente llegaba a su momento de más pronuniciación, lo que no me parece realista - cuando un organismo está en la mitad de su vida, no es cuando más ratio de vitalidad pierde. También estuve investigando y probando otras funciones que fueran alternativas, descubrí recursos muy útiles como este. No obstante, encontré varias funciones de **easing** para transiciones suaves muy utilizadas en animación:

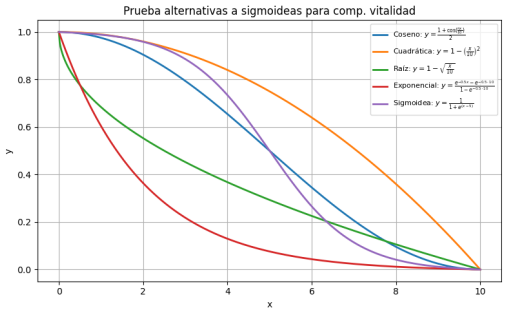


Figura 2.14: Coseno normalizado, cuadrática descendente, raíz cuadrada invertida, decaimiento exponencial escalado y logística (sigmoidea)

Las funciones del coseno normalizado y cuadrácticas descendiente, querían parecerse las que más a lo que estaba buscando. Pero durante el proceso descubrí el **modelo de crecimiento de Gompertz** (*Gompertz, 1825*[18]), que modela el envejecimiento biológico. Me baso en ese modelo; pero en su forma espejo, ya que me interesa el deterioro y no crecimiento. Este modelo es utilizado incluso hoy en día para modelar la mortalidad en poblaciónes biológicas (*Golubev, 2009*[19]). La idea, la he orientado a establecer una esperanza de vida (lifespan) para una especie de flora/fauna y que, en condiciones ideales, cuando la entidad haya llegado a esa edad, es cuando su vitalidad llega a 0 y muere.

$$\lambda = 1 - e^{-\alpha \cdot e^{\beta \cdot p}} \tag{2.23}$$

Símbolo	Descripción
λ	Pérdida de vitalidad.
p	Proporción de vida consumida (A_{bio}/esperanza de vida).
α	Inicio del deterioro (valor usado: 0.005).
β	Pendiente de deterioro (valor usado: 7.0).

Produce una curva sigmoidea pero más maleable y donde la tasa de pérdida de vitalidad se acelera en las etapas avanzadas de la vida - esto es lo bonito y además, lo que necesito; captura de manera muy elegante el incremento exponencial de la mortalidad con la edad y, es sabido que es un fenómeno común en muchas especies (*Vaupel et al., 1998*[20]). Veamos el efecto con diferentes tasas de envejecimiento (que ayuda a calcular la edad biológica):

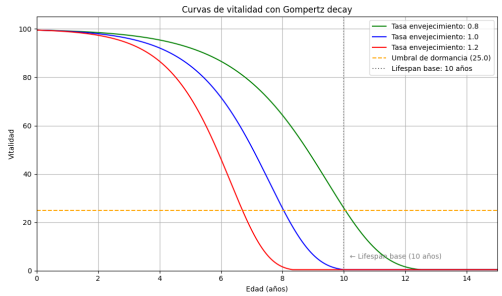


Figura 2.15: Gompertz decay

Durante la simulación, existirán multitud de factores externos que harán que la tasa de envejecimiento fluctúe, con lo que es muy complicado que la vitalidad de la entidad llegue a 0 cuando se complete el tiempo que dicte el lifespan. **A menor tasa de envejecimiento, mayor proyección de longevidad**.

2.4.1.3. Gestión del estrés

En el componente vital mantengo un nivel de estrés que se incrementa por factores ambientales adversos (clima, por ejemplo) y se reduce durante condiciones óptimas:

$$S_{new} = S_{current} + \Delta S \tag{2.24}$$

Símbolo	Descripción
S_{new}	Estrés tras la actualización.
$S_{current}$	Estrés actual antes de la actualización.
ΔS	Incremento de estrés.

El estrés acumulado afecta directamente a la vitalidad y acelera el deterioro natural. Existen varias fuentes de estrés: **temperatura, hambre, sed, falta de energía, etc**.

Dependiendo un poco del umbral de vitalidad se hace una pequeña penalización por estrés; a más baja esté la vitalidad, mayor penalización. Con esto, lo que quiero representar es un **deterioro de la capacidad de gestión del estrés conforme la edad avanza**.

Efecto de hormesis: cómo afecta el estrés a la tasa de envejecimiento

Investigando sobre cómo afecta el estrés, aprendí sobre el **fenómeno de hormesis** - donde dósis bajas de estrés son positivas para el organismo (*Calabrese y Blain, 2005*[21]). He modelado esta idea para calcular la tasa de envejecimiento: si la entidad tiene menos de 0.3 de estrés, la tasa de envejecimiento será baja, pero si es mayor, comenzará a empeorar.

Entonces, he diseñado la tasa para que un valor de 1.0 se de al alcanzar exactamente el lifespan definido para la especie. Valores inferiores proyectan una vida más larga y superiores la acortan. No encontré un valor estándar para determinar cuándo el estrés según la hormesis deja de ser beneficioso, así que establecí 0.3 como umbral; depende del organismo y situación.

Tampoco conseguí encontrar una función específica, así que pensé en usar **funciones por tramos definiendo yo los puntos**. Quería que al inicio de la vida, donde inevitablemente hay estimulación externa y estrés, fuera positivo. Pero una vez superado el umbral de 0.3, la tasa subiera rápidamente al valor base de 1.0, y después acelerarara hasta un máximo de 1.5 con el 100 % de estrés.

Tramo 1: $s \in [0,0,3]$ **Tramo 2:** $s \in [0,3,1,0]$

$$f(s) = \begin{cases} a_0 + a_1 s + a_2 s^2 + a_3 s^3, & 0 \le s \le 0{,}3, \\ b_0 + b_1 s + b_2 s^2 + b_3 s^3, & 0{,}3 \le s \le 1, \end{cases} \quad \begin{aligned} f_1(0) &= 1, & f_1'(0) &= 0, \\ f_1(0{,}3) &= 0{,}85, & f_1'(0{,}3) &= 0, \\ f_2(0{,}3) &= 0{,}85, & f_2'(0{,}3) &= 0, \\ f_2(1) &= 1{,}5, & f_2'(1) &= 0. \end{aligned} \tag{2.25}$$

Cada tramo tiene 4 condiciones (valor y derivada en extremos), por lo que se puede modelar con un polinomio cúbico (grado 3), con lo que monté el sistema y utilicé `np.linealg.solve` para resolverlo (sistema en comentarios). Con los coeficientes ya pude forjar la función final y programarla. Con ello ya puedo calcular el valor de la **tasa de envjecimiento en función del estrés**:

$$\mathcal{A}(S) = \begin{cases} 1{,}0 - 5{,}0\,S^2 + 11{,}111111111111109\,S^3, & S \le 0{,}3, \\ 1{,}3104956268221577 - 3{,}4110787172011685\,S + 7{,}390670553935865\,S^2 - 3{,}790087463556854\,S^3, & S > 0{,}3. \end{cases} \tag{2.26}$$

Donde S es el estrés normalizado (estrés actual / estrés máximo). Veamos el resultado final:

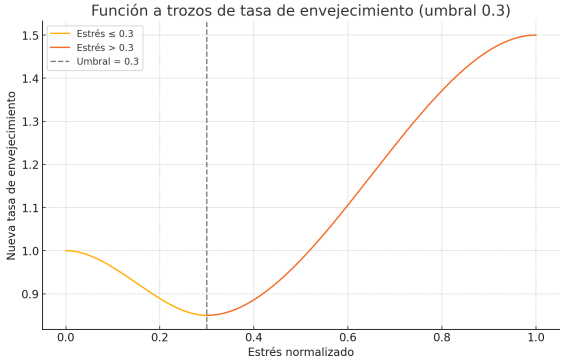


Figura 2.16: Hormésis a trozos custom

2.4.1.4. Integridad somática

La integridad somática representa la salud física inmediata, me inspiré en teorías de acumulación de daño y límite de Hayflick en senescencia celular **k**. He encontrado estas referencias, pero en mi

caso, aparenta ser más de lo que es - es simplemente un método para controlar los *puntos de vida*, como en videojuegos. Puede reducirse por daño físico (depredación, condiciones extremas, etc.) y recuperarse según la tasa de regeneración. Si cae a cero, la entidad muere, sin importar su vitalidad.

2.4.2. Componente fotosintético

La función principal de este componente es *emular* un sistema metabólico para la flora – **gestiona tanto producción como consumo de energía mediante fotosíntesis**. Se centra en calcular y actualizar eficiencia fotosintética, tasa de respiración, actividad metabólica y reservas energéticas. Como he mencionado antes, he trazado un diseño simplista, pero pese a ello he intentado integrar factores limitantes que predominan en ecofisiología vegetal (*Lambers et al., 2008*[22]). Por ejemplo, me he basado en la **Ley del Mínimo de Liebig**, que dice que el crecimiento está limitado por el recurso más escaso, junto con principios de co-limitación (*Bloom et al., 1985*[23]). Es una idea que he tenido muy presente a lo largo del proyecto; lo veremos con multitud de multiplicaciones de factores normalizados.

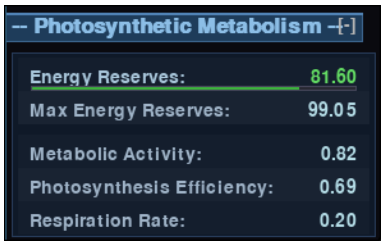


Figura 2.17

2.4.2.1. Cálculo de producción energética

En el cálculo influyen múltiples factores; tanto ambientales como fisiológicos. Con el siguiente cálculo obtenemos un factor, que describe a **efectos prácticos y, para un instante dado, la fotosíntesis efectiva**:

$$P_{eff} = E_{photo} \cdot L_{avail} \cdot M_{temp} \cdot M_{water} \cdot A_{metab} \cdot F_{dorm} \quad (2.27)$$

$$R_{eff} = R_{base} \cdot A_{metab} \cdot F_{resp} \quad (2.28)$$

Símbolo	Descripción
P_{eff}	Fotosíntesis efectiva.
E_{photo}	Eficiencia fotosintética **base**.
L_{avail}	Disponibilidad de luz.
M_{temp}	Modificador por temperatura.
M_{water}	Modificador por agua.
A_{metab}	Actividad metabólica.
F_{dorm}	Fac. dormancia fotos. (0.15 si dormante, 1.0 si no).
R_{eff}	Respiración efectiva.
R_{base}	Tasa de respiración base.
F_{resp}	Fac. dormancia resp. (0.1 si dormante, 1.0 si no).

Todos ellos están normalizados y, algunos como la disponibilidad de luz y el modificador de absorción de agua no disponen aún de lógica implementada, con lo que sus valores son de 1.0 hasta que sean extendidos.

Por otro lado, la tasa de **respiración efectiva** la calculo de manera similar pero con diferente impacto si la planta está en **estado de dormancia** (estado de emergencia que activa la flora y vive con mínimo gasto durante un tiempo).

De nuevo, todos ellos son valores normalizados que determinan con qué intensidad se da ese parámetro. Nótese que los valores base, como se verá más adelante, **forman parte del fenotipo**. El factor de dormancia, lo he puesto para que en el caso de estar dormante la respiración efectiva de la planta sea mucho menor.

2.4.2.2. Eficiencia metabólica y balance energético

La eficiencia metabólica la calculo como ratio, esto es; en función de la proporción entre respiración y fotosíntesis **efectivas**, comparada con una ratio óptimo predefinida:

$$R_{current} = \frac{R_{eff}}{P_{eff}} \tag{2.29}$$

$$E_{metab} = 1{,}0 - \frac{|R_{current} - R_{opt}|}{R_{opt}} \cdot 1{,}5 \tag{2.30}$$

Símbolo	Descripción
$R_{current}$	Ratio actual entre respiración y fotosíntesis.
R_{opt}	Ratio óptimo entre resp. y fotos. (he escogido 0.2 por defecto).
E_{metab}	Eficiencia metabólica resultante.

El 1.5, es un factor de escalado o amplificación que he introducido manualmente para intentar que afecte más a desviaciones grandes del óptimo (quiero que, el término al que modula, **a más se desvíe, más caiga la eficiencia**).

Teniendo ya estos ratios, podemos calcular el **delta energético base** - se calcula a con la diferencia entre fotosíntesis efectiva y respiración efectiva, modulado por la eficiencia metabólica. **La lógica aquí es**; la fotosíntesis produce - la respiración gasta, calculamos su diferencia y la escalamos por lo bien que funcione el sistema metabólico de la entidad.

$$\Delta E_{base} = (P_{eff} - R_{eff}) \cdot E_{metab} \tag{2.31}$$

Con esto consigo una **eficiencia metabólica variable**; las plantas alcanzan máxima eficiencia cuando la relación respiración/fotosíntesis está en su valor óptimo; me baso en la relación de producción/gasto entre fotosíntesis/respiración (*Van Oijen et al., 2010*[24]) y en el análisis de (*Amthor, 2000*[25]) que habla sobre que una proporción R/P de alrededor de 0.2 suele ser típica en plantas sanas; menos de ese umbral, en general la fisiología no se sostiene - sin respiración suficiente, muchos procesos empiezan a fallar (mantenimiento celular, producción de biomasa...etc).

Que el envejecimiento sea un factor que reduce las reservas energéticas y hace que el metabolismo sea menos eficiente, no me ha sorprendido - es algo intuitivo que además está demostrado (*Niinemets, 2002*[26]). Sí que lo ha hecho un factor que no conocía en este contexto; **pérdidas por entropia**. Este tipo de pérdidas son la segunda ley de la termodinámica en sistemas biológicos y, al parecer, es un fenómeno inevitable en todos los procesos metabólicos.

Básicamente es un factor que aumenta el desorden en los sistemas vivos y requiere constante energía para contrarrestarla (*Schneider y Kay, 1994*[27]).

$$\Delta E_{entropy} = 0{,}02 \tag{2.32}$$

$$F_{age} = 1{,}0 + \left(\frac{A_{bio}}{L_{span} \cdot \tau_{year}}\right)^2 \cdot 0{,}5 \tag{2.33}$$

$$\Delta E_{age} = E_{current} \cdot 0{,}002 \cdot F_{age} \tag{2.34}$$

Símbolo	Descripción
$\Delta E_{entropy}$	Decaimiento por entropía.
F_{age}	Factor de edad.
A_{bio}	Edad biológica.
L_{span}	Esperanza de vida.
τ_{year}	Número de ticks por año.
ΔE_{age}	Decaimiento por edad.
$E_{current}$	Reservas energéticas actuales.

En F_{age} tenemos el término de edad relativa (A_{bio} está en años; el valor de ese ratio vale 0 al nacer y 1 cuando llega al lifespan en años). Intento modelar precisamente lo que se comentaba sobre la relación entre el envejecimiento y el la eficiencia metabólica; a más edad, mayor será el factor de descuento que restaremos, por eso elevo al cuadrado, para expresar ese crecimiento y no linealidad. El 1.0 es para que, nada más nacer, el factor no afecte ya que 1.0 + 0 = 10 y, el 0.5, como en otras ocasiones, lo utilizo para modular.

El cambio energético final lo calculo simplemente restando estos factores de decaimiento al cambio energético base:

$$\Delta E_{final} = \Delta E_{base} - (\Delta E_{entropy} + \Delta E_{age}) \tag{2.35}$$

2.4.2.3. Gestión del estrés metabólico

También compruebo constantemente el nivel de reservas de energia de la entidad, y doy una pequeña penalización de estrés en función de varios umbrales posibles:

$$\Delta S = \begin{cases} S_{no\_energy}, & \text{si } E_{factor} \leq 0{,}0005 \\ S_{critical}, & \text{si } 0 < E_{factor} < E_{critical} \\ S_{low}, & \text{si } E_{critical} \leq E_{factor} < E_{low} \\ S_{sufficient}, & \text{si } E_{sufficient} < E_{factor} \leq E_{abundant} \\ S_{abundant}, & \text{si } E_{factor} > E_{abundant} \end{cases} \tag{2.36}$$

Efecto del estrés en la eficiencia fotosintética:

Al igual que vimos en el componente vital, que el estrés influía en sus factores, aquí también hago que tenga un impacto sobre la eficiencia fotosíntética. He utilizado un modelo que vuelve a contemplar la hormesis, pero ésta vez adaptada a éste componente.

En el componente vital, quería algo un poco específico para la tasa de crecimiento, sin embargo ahora me he aventurado a investigar sobre diferentes funciones que tengan un máximo intermedio (*Greenwood et al., 2022*[28]), pero me parecían excesivamente complejas para modelar lo que busco.

Entonces, para dar variabilidad, he decidido un umbral diferente - además me he apoyado en la teoría de que, **para cada proceso, la curva de hormesis es diferente**. En este caso he sido más estricto y he reducido el umbral al 25 % del máximo. Según (*Calabrese y Blain, 2005*[21]), los valores de dosis que estimulan vs. inhiben varían para cada proceso; recopilaron 5.600 respuestas horméticas en más de 1.450 estudios, y concluyen que el rango depende del modelo biológico.

Con lo que, para construir la función, veamos las condiciones que necesito:

- Cuando stress es 0, eficiencia 1, esto es, nivel basal: $F(0) = 1$
- Incremento hormético: $F(0{,}25) = 1{,}25$
- Punto óptimo de estrés: $s_{opt} = 0{,}25$
- Para máximo, derivada nula en pico: $F'(0{,}25) = 0$
- Valor nulo al final: $F(1) = 0$

$$F_1(s) = \alpha_0 + \alpha_1\, s + \alpha_2\, s^2, \qquad 0 \le s \le 0{,}25,$$

$$F_2(s) = \beta_0 + \beta_1\, s + \beta_2\, s^2, \qquad 0{,}25 \le s \le 1.$$

Elijo cada tramo cuadrático porque busco que cada tramo satisfaga **tres restricciones linealmente independientes**:

1. Valor en el extremo del tramo: $F_1(0) = 1$ ó $F_2(1) = 0$
2. Valor en el pico: $F_1(0{,}25) = F_2(0{,}25) = 1{,}25$
3. Derivada nula en el pico: $F_1'(0{,}25) = F_2'(0{,}25) = 0$

Con lo que, para cubrir 3 ecuaciones necesito un polinomio de grado dos, ya que un polinomio de grado n tiene $n+1$ coeficientes libres. En este caso necesito 3, por tanto, un grado 2 basta. De nuevo con `np.linealg.solve`:

$$F(s) = \begin{cases} 1{,}25 - 4\,(s - 0{,}25)^2, & 0 \le s \le 0{,}25, \\ 1{,}25 - \dfrac{20}{9}\,(s - 0{,}25)^2, & 0{,}25 \le s \le 1. \end{cases}$$

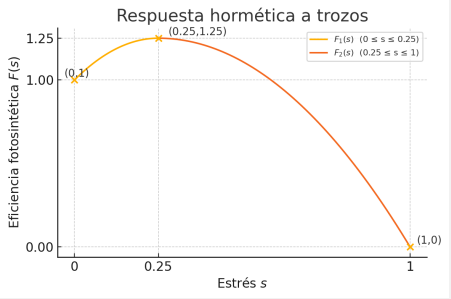


Figura 2.18: Curva de eficiencia $F(s)$ según nivel de estrés s

Este valor que nos devuelve la función para un determinado estrés, es un factor que aplicamos a la fotosíntesis efectiva. Por ejemplo, si nos fijamos en la imagen, cuando el estrés esté en su mejor estado, el 0.25, el factor por el que multiplicaremos la eficiencia fotosintética actual será de 1.25, el máximo.

$$P_{new} = P_{eff} \times F(S_{norm}) \tag{2.37}$$

Símbolo	Descripción
S_{norm}	Estrés normalizado (estrés actual / estrés máximo).
F_{adapt}	Factor de adaptación.
P_{eff}	Eficiencia fotosintética.
P_{new}	Nueva eficiencia fotosintética ajustada.

A más estrés, mayor tasa de respiración; modelo su impacto con un patrón cuadrático creciente:

$$I_{resp} = 1 + S_{norm}^2$$

$$R_{new} = \text{mín}(0{,}2,\, R_{base} \cdot I_{resp}) \tag{2.38}$$

Símbolo	Descripción
I_{resp}	Impacto del estrés en la respiración
R_{base}	Tasa de respiración base.
R_{new}	Nueva tasa de respiración.

Al igual que antes, el 1 está porque si 0 stress, el impacto será 1.0 (inocuo al multiplicar).

2.4.3. Componente de nutrición autótrofa

Vamos ahora con un intento un poco *naïve* de implementar un sistema de nutrición autótrofa; con este componente modelo la interfaz flora-suelo y algunas relaciones simbióticas que ocurren bajo tierra, con objeto de calcular cómo gana energía la flora absorbiendo nutrientes - autoabasteciéndose:

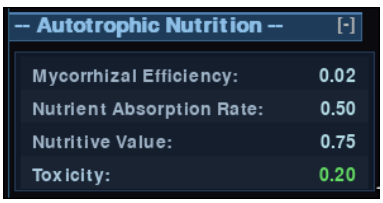


Figura 2.19

$$\alpha_{abs} = \beta_{base} \cdot \eta_{stress} \cdot (\gamma_{tox} + \epsilon) \cdot \xi \tag{2.39}$$

$$\gamma_{tox} = (1{,}0 - \tau) \cdot 0{,}7 \tag{2.40}$$

$$\Delta E = \alpha_{abs} \cdot E^{máx} \tag{2.41}$$

Símbolo	Descripción
α_{abs}	Tasa de absorción efectiva.
β_{base}	Tasa base de absorción ($\beta_{base} = \nu_{abs} \cdot 0{,}01$).
ν_{abs}	Parámetro de absorción de nutrientes (0.3 por defecto).
η_{stress}	Ratio de estrés ($\eta_{stress} = 1 - \frac{stress}{maxstress}$).
γ_{tox}	Factor de modificación por toxicidad.
τ	Nivel de toxicidad actual.
ϵ	Constante pequeña para no propagar 0 (tox. = 0 es posible).
ξ	Variabilidad ambiental (distr. uniforme $\mathcal{U}(0{,}9, 1{,}1)$).
ΔE	Ganancia energética (escala si $E^{máx}$ evoluciona).
$E^{máx}$	Reservas energéticas máximas.

De nuevo, uso factores multiplicativos para calcular en función de valores base una propiedad *efectiva*. Con esto expreso la idea de la ley de los Mínimos de Leibig - el estrés tiene un papel importante y lo he definido **inversamente proporcional a la tasa de absorción**. Esta dinámica es precisamente el concepto de asignación óptima de recursos que describe (*Bloom et al., 1985*[23]).

2.4.3.1. Actividad micorrícica

He aprendido una vez más sobre algo que no conocía, y he disfrutado mucho de ello; las relaciones micorrícicas - es una simbiosis entre plantas y hongos para optimizar la adquisición de nutrientes, que ocurre naturalmente: los hongos crecen hacia la planta cuando unos filamentos de raíz detectan señales químicas (*Johnson, 2010*[29]).

Para calcularlo, hago el beneficio micorrícico inversamente proporcional al estado energético de la planta, con la idea de **potenciar la *ayuda*** cuando más la necesita, ya que no he implementado sistemas de químicas de suelo:

$$\mu_{bonus} = \rho_{myc} \cdot \lambda_E \tag{2.42}$$

$$\lambda_E = 1{,}0 - \frac{E}{E^{máx}} \tag{2.43}$$

Símbolo	Descripción
μ_{bonus}	Beneficio energético de la actividad micorrícica.
ρ_{myc}	Tasa de eficiencia micorrícica.
λ_E	Factor de déficit energético.
E	Reservas energéticas actuales.
$E^{máx}$	Capacidad máxima de reservas energéticas.

2.4.3.2. Dinámica de toxicidad

Para la toxicidad - la cual está relacionada con los metabolitos secundarios que las plantas producen como defensa o como resultado del estrés metabólico - he tomado el enfoque de modelarla tal que esté influenciada por cambios en la eficiencia fotosintética:

$$\Delta\tau = \begin{cases} -\kappa_1 \cdot (\phi_{new} - \phi_{old}), & \text{si } \phi_{new} > \phi_{old} \\ \kappa_2 \cdot (\phi_{old} - \phi_{new}), & \text{si } \phi_{new} < \phi_{old} \end{cases} \tag{2.44}$$

$$\tau' = \tau + \Delta\tau \tag{2.45}$$

Símbolo	Descripción
$\Delta\tau$	Cambio en toxicidad.
κ_1	Factor de reducción de toxicidad (0.02).
κ_2	Factor de aumento de toxicidad (0.04).
ϕ_{new}	Nueva eficiencia fotosintética.
ϕ_{old}	Antigua eficiencia fotosintética.
τ	Nivel de toxicidad actual.
τ'	Nivel de toxicidad actualizado.
τ_{base}	Nivel base de toxicidad.

La toxicidad está relacionada al nivel de estrés también, así que hago que pueda cambiar aleatoriamente con ello, para introducir variabilidad:

$$\theta_{\text{stress}} = \theta_0 + \theta_1 \cdot \frac{\sigma}{\sigma^{\text{máx}}} \tag{2.46}$$

$$P(\text{inc}) = I(r < \theta_{\text{stress}}) \tag{2.47}$$

$$\Delta\tau_{\text{random}} = \begin{cases} \mathcal{U}(\delta^{+}_{\text{mín}}, \delta^{+}_{\text{máx}}) \cdot (1{,}0 + \eta_{\text{stress}}), & \text{si } P(\text{inc}) = 1 \\ -\mathcal{U}(\delta^{-}_{\text{mín}}, \delta^{-}_{\text{máx}}) \cdot (1{,}0 - \eta_{\text{stress}}), & \text{si } P(\text{inc}) = 0 \end{cases} \tag{2.48}$$

Símbolo	Descripción
θ_{stress}	Umbral de estrés para cambio de toxicidad.
θ_0	Parámetro base (0.3).
θ_1	Factor de escala (0.4).
r	Número aleatorio uniforme entre 0 y 1.
I	Función indicadora (1 si condición verdadera, 0 si no).
$P(\text{inc})$	Probabilidad de incremento de toxicidad.
$\delta^{+}_{\text{mín}}, \delta^{+}_{\text{máx}}$	Rango para aumento de toxicidad (0.005, 0.015).
$\delta^{-}_{\text{mín}}, \delta^{-}_{\text{máx}}$	Rango para disminución de toxicidad (0.003, 0.008).

La intuición que he utilizado en los mecanismos de toxicidad, está basada en la hipótesis de balance carbono-nutrientes (*Bryant et al., 1983*[30]); que dice que los compuestos tóxicos aumentan cuando el carbono excede las necesidades de crecimiento por limitaciones en otros nutrientes - por eso, asocio al déficit de producción de energía una ligera penalización. También, con la asimetría en las tasas ($\kappa_1 = 0{,}02$ vs $\kappa_2 = 0{,}04$) intento expresar una mayor facilidad para acumular toxinas que para eliminarlas, lo cual se menciona en estudios toxicológicos (*Straalen y Roelofs, 2006*[31]).

En estos momentos no está implementado pero, en este punto, algo que queda pendiente de implementar es que también se vea afectada por el estrés cuando un herbívoro se alimente de ella. Es bastante sencillo y rápido de implementar, pero como todo; lo que no es tanto es el hacer las pruebas. Queda pendiente.

2.4.4. Componente de nutrición heterótrofa

2.4.4.1. Dinámica del metabolismo

Ahora implemento un modelo de nutrición heterótrofa para representar la **dinámica de alimentación y consumo de agua de la fauna**. En este caso, para poder nutrirse, requieren de ingesta de **nutrientes externos**. Principalmente gestiona los niveles de hambre y sed, pero, como se puede observar a continuación, también calcula penalizaciones energéticas y niveles de estrés asociados:

Figura 2.20

$$H' = \text{máx}(0{,}0, H - r_h) \tag{2.49}$$

$$T' = \text{máx}(0{,}0, T - r_t) \tag{2.50}$$

$$r_h^{\text{eff}} = r_h \cdot \left(1{,}0 + \frac{S}{S^{\text{máx}}} \cdot 0{,}5\right) \tag{2.51}$$

$$r_t^{\text{eff}} = r_t \cdot \left(1{,}0 + \frac{S}{S^{\text{máx}}} \cdot 0{,}5\right) \tag{2.52}$$

Símbolo	Descripción
H	Nivel de hambre actual **(H=100 sin hambre)****
H'	Nivel de hambre actualizado.
r_h	Tasa base de hambre.
r_h^{eff}	Tasa efectiva de hambre.
T	Nivel de sed actual.
T'	Nivel de sed actualizado.
r_t	Tasa base de sed.
r_t^{eff}	Tasa efectiva de sed.
S	Nivel de estrés actual.
$S^{\text{máx}}$	Nivel máximo de estrés.

Las tasas de hambre/sed, son unos pequeños valores que se van a ir descontando en cada instante de tiempo. **La idea principal** aquí, es que el estrés incrementa las necesidades fisiológicas, y esto, pasivamente, hará que para tener una mayor ganancia energética, se requiera de una mejor eficiencia metabólica (siguiente apartado). Esto es algo bastante intuitivo, pero pese a adoptar una vertiente simplista para los cálculos, he querido verificar que las nociones son correctas, y efectivamente, esto es muy similar a la que describe (*McEwen, 2000*[32]) sobre **alostasis**, donde el estrés eleva los requisitos metabólicos del organismo.

2.4.4.2. Consumo de alimentos

Modelo la ingesta de alimentos teniendo en cuenta tanto el valor nutritivo como el nivel actual de hambre - lo amplifico con un mecanismo de compensación, para que la eficiencia de conversión aumente con el nivel de hambre:

$$f_{hunger} = 5{,}0 + (100{,}0 - H) \cdot 0{,}25 \tag{2.53}$$
$$N_{amp} = N_{val} \cdot f_{hunger} \tag{2.54}$$
$$\Delta H = \text{mín}(H_{máx} - H, N_{amp}) \tag{2.55}$$
$$\Delta E = N_{amp} \cdot \eta_{met} \tag{2.56}$$

Símbolo	Descripción
f_{hunger}	Factor de amplificación por hambre.
N_{val}	Valor nutritivo del alimento consumido.
N_{amp}	Valor nutritivo amplificado.
ΔH	Incremento en nivel de hambre (siempre N_{amp}, salvo edge cases)
$H_{máx}$	Nivel máximo de hambre (100.0).
ΔE	Ganancia energética.
η_{met}	Eficiencia metabólica.

De esta manera consigo el fenómeno de la **hiperfagia compensatoria** (*Känkänen y Pirhonen, 2009*[33]; *Ali et al., 2003*[34]; *Skalski et al., 2005*[35]); esto es que un organismo con mayor déficit nutricional maximiza la eficiencia de absorción. El factor base de 5.0 es simplemente para cerciorarme de que incluso en estados de saciedad haya un mínimo.

****Nota:** Al principio, expresé en código tanto el hambre como la sed, como un factor de 0 a 100, donde 100 es que no tiene hambre y 10, que tiene muchísima. No es intuitivo, pero lo fuí dejando y así ha quedado.

2.4.4.3. Consumo de agua

Para la hidratación, aplico un modelo más simple pero también necesario para la homeostasis energética:

$$\Delta T = \text{mín}(T_{máx} - T, H_{val}) \tag{2.57}$$
$$\Delta E_{hydration} = H_{val} \cdot 0{,}3 \tag{2.58}$$

Símbolo	Descripción
ΔT	Incremento en nivel de sed.
$T_{máx}$	Nivel máximo de sed (100.0).
H_{val}	Valor de hidratación del agua consumida.
$\Delta E_{hydration}$	Impulso energético por hidratación.

El factor 0.3 para la conversión energética es un factor de ajuste que he introducido manualmente tras muchas pruebas, para estabilizar un poco los valores, aunque este valor debería de ser dinámico.

2.4.4.4. Penalizaciones energéticas

Si el estado metabólico / reservas energéticas están en un estado deteriorado, aplico penalización de energía:

$$P_{base} = r_h \cdot 0{,}3 + r_t \cdot 0{,}3 \tag{2.59}$$
$$P_{energy} = \begin{cases} 1{,}0 \cdot \exp\left(4{,}0 \cdot \left(1{,}0 - \frac{E/E_{máx}}{0{,}1}\right)\right), & \text{si } \frac{E}{E_{máx}} < 0{,}1 \\ 0, & \text{en otro caso} \end{cases} \tag{2.60}$$
$$P_{total} = P_{base} + P_{energy} \tag{2.61}$$

Símbolo	Descripción
P_{base}	Penalización energética base.
P_{energy}	Factor de estrés por energía baja.
E	Reservas energéticas actuales.
$E_{máx}$	Capacidad máxima de reservas energéticas.
P_{total}	Penalización energética total.

Nótese la penalización exponencial; ocurre solo en niveles críticos de energía (menos del 0.1) y me he basado en principios de respuesta a déficits energéticos severos - el 4.0 es simplemente un fruto de ajuste de pendiente
Realmente, lo que hago es un agregar un pequeño delta de estrés dependiendo de los cambios.

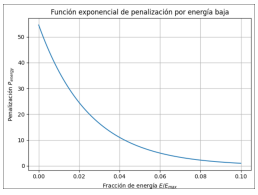


Figura 2.21

2.4.4.5. Dinámica de estrés

El estrés fisiológico lo modelo de manera simple; agrego un pequeño delta de estrés a la entidad dependiendo del estado del hambre o la sed (se analizan y aplica estrés por separado).

Además, **si la reserva energética de la entidad llega a 0**, se aplica una gran cantidad de estrés continuamente.

2.4.5. Componente de crecimiento

En éste módulo gestiono los cambios de tamaño, velocidad de desarrollo y transiciones entre fases de madurez. Analizaré ahora los mecanismos matemáticos que he implementado para capturar estas dinámicas.

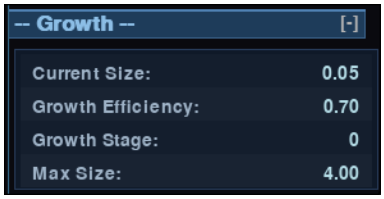


Figura 2.22

2.4.5.1. Progresión temporal

Lo primero que necesitamos calcular es un factor/ratio del avance temporal relativo; básicamente, dónde se encuentra la entidad dentro de su vida potencial:

$$R_{bio} = \frac{A_{current}}{L_{span}} \tag{2.62}$$

$$R_{bio\_mod} = R_{bio}\, M_{growth}\, E_{growth} \tag{2.63}$$

Símbolo	Descripción
E_{growth}	Eficiencia de crecimiento (factor ambiental y fisiológico).
R_{bio}	Ratio de edad biológica.
R_{bio_mod}	Ratio de edad biológica modificada.
M_{growth}	Modificador de crecimiento (factor evolutivo).
$A_{current}$	Edad actual (en ticks).
L_{span}	Esperanza de vida (en ticks).

R_{bio} es un ratio que por sí mismo no captura todo lo necesario; hay otros factores que pueden acelerar o retrasar el desarrollo, como el estrés, etc. Entonces, para poder introducir esta variabilidad, aplico modificadores que ajustan la tasa efectiva de crecimiento con R_{bio_mod}.

Con el modificador de crecimiento (M_{growth}) quiero expresar o representar las características inherentes a la especie e individuo; quiero conseguir que las adaptaciones evolutivas aceleren o retrasen el desarrollo. Por otro lado, la eficiencia (E_{growth}) me fluctúa constantemente, hago que se vea afectada por factores ambientales externos; entre ellos, el estrés.

2.4.5.2. Transformación no lineal de crecimiento

La forma en la que inicialmente me planteé lo que deseaba, era: **"Si mi edad biológica (no cronológica) ha llegado al 70 % de lo que puede llegar, ¿cuanto he crecido del máximo?"** - ese problema es el que intento resolver con G_{ratio}.

Con lo que, aquí radica la esencia de este componente; el crecimiento orgánico no es lineal, así que, después de analizar múltiples modelos biológicos, decidí implementar una simple **transformación sigmoidea**; ya que aunque no describa perfectamente el crecimiento, me sirve para modelar: **latencia inicial, expansión rápida y saturación final**:

$$G_{ratio} = \frac{1}{1 + e^{-k(R_{bio\_mod} - 0{,}5)}} \tag{2.64}$$

Símbolo	Descripción
G_{ratio}	Ratio de completitud de crecimiento.
k	Factor de inclinación de la curva (**valor 6**).

La elección del parámetro k **no es arbitraria**; he experimentado con varios valores entre 2 y 12. Valores bajos ($k < 4$) generan curvas poco pronunciadas (casi lineal, de hecho) y, valores excesivos ($k > 8$) transiciones demasiado súbitas - aunque no busque exactitud, me parecen bastante alejadas de contextos biológicos.

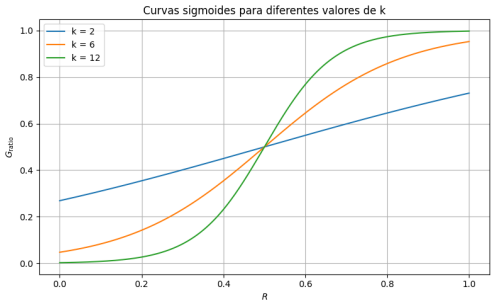


Figura 2.23

2.4.5.3. Proyección dimensional

Ahora que ya tenemos la proporción de crecimiento posible completado, ya puedo calcular el tamaño y actualizarlo. Para ello computo la interpolación entre dimensiones iniciales y máximas:

$$S_{new} = S_{init} + (S_{max} - S_{init}) \cdot G_{ratio} \tag{2.65}$$

Símbolo	Descripción
S_{new}	Nuevo tamaño calculado.
S_{init}	Tamaño inicial (semilla/plántula en flora, por ejemplo).
S_{max}	Tamaño máximo genéticamente determinado.
G_{ratio}	Ratio de completitud de crecimiento.

La distribución de tamaños máximos iniciales es por especie, pero introduzco un pequeño ruido aleatorio en la creación de cada individuio - no obstante, forma parte de los genes de tanto al flora como la fauna, así que formará parte de la evolución.

2.4.5.4. Discretización de fases ontogenéticas

El crecimiento es un proceso continuo, pero también existen algunas propiedades que aparecen o emergen en ciertas fases del desarrollo de un individuo; floración, fructificación, cambios estructurales etc. Así que implementé un sistema de etapas definidas por umbrales:

$$T_i = S_{max} \cdot \frac{i}{N_{stages}} \tag{2.66}$$

Símbolo	Descripción
T_i	Umbral dimensional para la etapa i.
S_{max}	Tamaño máximo potencial.
N_{stages}	Número total de etapas ontogenéticas.

He consultado literatura sobre morfogénesis vegetal y animal, y el estándar tiende a ser el representarlo entre 4-6 fases, depende un poco de la fuente - con lo que decidí establecerlo en 4.

2.4.5.5. Modulación por estrés ambiental

El estrés también impacta al crecimiento; bajo condiciones adversas, los recursos que normalmente se centran el crecimiento, se redireccionan a supervivencia.

$$E_{growth}^{(t+1)} = E_{growth}^{(t)}(1 - 0{,}6\,S_n) \tag{2.67}$$

Símbolo	Descripción
E_{growth}	Eficiencia de crecimiento ajustada.
S_n	Estrés normalizado (escala 0–1).

El coeficiente 0.6 ha sido ajustado manualmente - no quiero que sea 1:1, el 60 % parece un buen balance y no quiero que se detenga completamente el crecimiento si el estrés es máximo - con ello quiero también expresar la sensibilidad al estrés; valores mayores para hipersensibilidad y menores para mayor tolerancia. También se podría hacer dinámico este factor para que sea particular al individuo e incluso se vea afectado por otros factores.

2.4.6. Componente de adaptación climática

Llegamos al último componente (existen más, pero están relacionados a transformaciones, movimientos etc), con él intento modelar la respuesta de las entidades a fluctuaciones en factores climáticos para que gestione interaccinoes con temperatura, eventos meteorológicos etc. Como vamos a ver, difiere un poco con respecto a los vistos hasta ahora - funciona casi exclusivamente de forma reactiva ya que se centra en responder a eventos ambientales que el bioma genera.

Figura 2.24

Para desarrollarlo, he investigado un poco al respecto y me he basado en principios ecofisiológicos sobre rangos de tolerancia térmica (*Larcher, 2003*[36]) - es por ello que he establecido para **cada especie una óptimo de temperatura** y, la lógica del componente se centra en gestionar las desviaciones del mismo, con ello, una vez calculadas, se genera una cantidad de estrés térmico que es proporcional a la magnitud de la desviación, pero además **modulado por las resistencias** (al frio y al calor).

2.4.6.1. Mecanismos de resistencia térmica

He representado las resistencias de manera asimétrica; esto es, se pueden tener distintos niveles de tolerancia a temperaturas frías y cálidas:

$$R_{selected} = \begin{cases} R_{cold}, & \text{si } T < T_{opt} \\ R_{heat}, & \text{si } T \geq T_{opt} \end{cases} \tag{2.68}$$

Símbolo	Descripción
$R_{selected}$	Resistencia térmica aplicable.
R_{cold}	Resistencia a temperaturas frías.
R_{heat}	Resistencia a temperaturas cálidas.
T	Temperatura ambiental actual.
T_{opt}	Temperatura óptima para la entidad.

La intuición en la que me he basado es simple: muchas especies mediterráneas, por ejemplo, tienen alta tolerancia al calor pero baja resistencia a heladas, y a su vez especies alpinas pueden tener el patrón inverso. $R_{selected}$ es la resistencia con la que vamos a modular la cantidad de estrés.

2.4.6.2. Cálculo de desviación térmica

Lo primero es saber el impacto fisiológico - para ello calculo cuánto se aleja la temperatura actual del óptimo:

$$\Delta T = T - T_{opt} \tag{2.69}$$

Símbolo	Descripción
ΔT	Desviación térmica.
T	Temperatura ambiental actual.
T_{opt}	Temperatura óptima para la entidad.

Con el signo identifico si estamos ante un estrés por frío ($\Delta T < 0$) o por calor ($\Delta T > 0$), esto me sirve para, *activar el mecanismo* de resistencia que corresponda en el momento.

2.4.6.3. Amplitud de tolerancia térmica adaptativa

He utilizado en diferentes ocasiones distribuciones gaussianas y **dado el cómo sigma controla la amplitud/anchura de la campana**, lo que pensé es en intentar hacer que sigma fuera dinámica, para que se pudiera adaptar a la resistencia del individuo. Entonces, si hacía que la resistencia modulase sigma y con ello, la tolerancia a la temperatura - iba a conseguir que entidades con mayor resistencia soportaran mucho mejor el estrés, pero, también ampliasen su zona de confort térmico:

$$\sigma_{\text{eff}} = \frac{\sigma_{\text{base}}}{1{,}0 - R_{\text{selected}}}$$

Símbolo	Descripción
σ_{eff}	Amplitud efectiva de tolerancia térmica.
σ_{base}	Amplitud base (15.0 por defecto, tras múltiples pruebas).
R_{selected}	Resistencia térmica aplicable (al frío o al calor)

Así lo implementé, pero la función que terminé escogiendo tiene forma y curva hiperbólica; para que, cuanto más se aproxime la resistencia a 1.0 - **inmunidad completa** - la amplitud de tolerancia tiende a infinito al haber asíntota. Entonces, cuando $R_{\text{selected}} \geq 1{,}0$, considero que la entidad posee inmunidad total a ese tipo de estrés térmico.

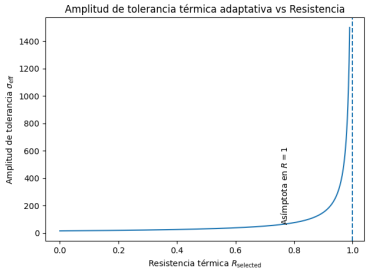


Figura 2.25

2.4.6.4. Modelado del estrés térmico mediante función gaussiana

Como he mencionado, para calcular el factor de estrés he utilizado una gaussiana - por un lado puedo modular la tolerancia en función de la resistencia, y **centro la distribución en la temperatura óptima de la entidad** - es perfecta para mi caso. Entonces, con estos parámetros, lo puedo ajustar para cada individuo y puedo calcular el **delta del estrés en función de la desviación con respecto a la temperatura actual**. La idea es que el estrés aumente al alejarse del óptimo, y además con mayor pendiente cuanto mayor es la desviación:

$$\Delta\text{stress} = \exp\left(-\frac{\Delta T^2}{2 \cdot \sigma_{\text{eff}}^2}\right) \tag{2.70}$$

Símbolo	Descripción
Δstress	Delta de estrés térmico a aplicar.
ΔT	Desviación térmica (Temperatura actual − Temperatura óptima).
σ_{eff}	Amplitud efectiva de tolerancia.

Obsérvese que Δstress es mínimo cuando $\Delta T = 0$, y decrece exponencialmente conforme aumenta la desviación σ. Con ello genero una campana de tolerancia alrededor del óptimo térmico, cuya anchura depende de σ_{eff}.

Parémonos para un ejemplo rápido; supongamos que una entidad tiene **temperatura óptima 20ºC**, y **resistencia al calor 0.05**, en un bioma donde en un instante determinado, la **temperatura es de 32º**. Agrego, a modo de comparativa, otra resistencia al calor de 0.5 para mostrar:

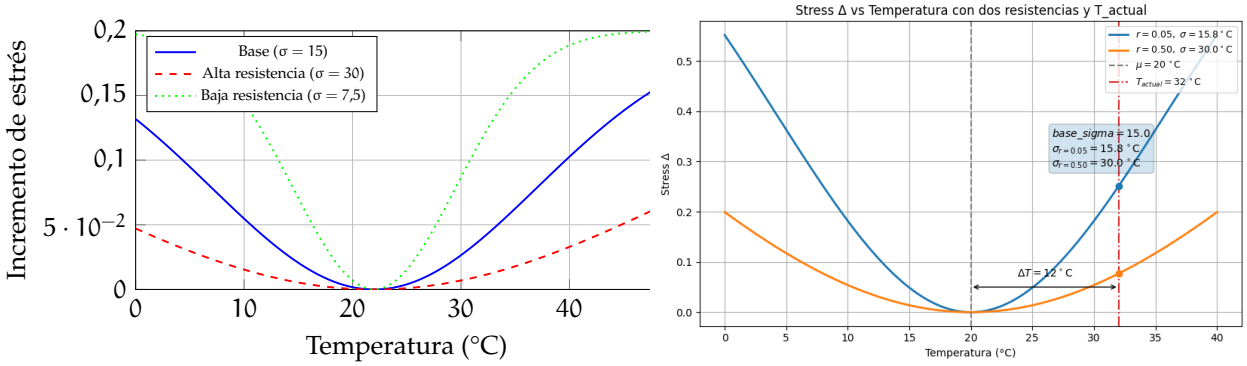


Figura 2.26: (a) Curva Gauss. de estrés vs temp. (b) e.g: clima tropical (≈30º, y entidad con temp. ópt 20º)

2.4.6.5. Alivio térmico y recuperación

Para que no sea tiranía y no todo sean penalizaciones, he agregado un factor de alivio de estrés cuando las condiciones son favorables; si la temperatura es muy cercana al óptimo ($\Delta S \leq 0{,}001$),

aplico una reducción de estrés mínima, un pequeño alivio:

$$\Delta S_{\text{relief}} = -C_{\text{optimal}} \tag{2.71}$$

Símbolo	Descripción
ΔS_{relief}	Reducción de estrés por condiciones óptimas.
C_{optimal}	Constante de alivio.

En un principio lo hice para ajustar un poco los cálculos, pero lo he dejado ya que, si se mira desde otra perspectiva, se podría decir que cuando las condiciones meteorológicas son favorables, las entidades tienen un proceso de restauración fisiológica.

2.4.7. Sistemas de componentes: sistema descentralizado vs centralizado

Como se puede observar, en el código existen por un lado el componente y por otro, su gestor o manager. Ésta versión no fué la primera, esto es una versión optimizada y fué fruto de una refactorización (Ver código de refactorización) que hice para mejorar el rendimiento (el indicado es el commit base, luego se siguieron mejorando y adaptando el resto de componentes).

Inicialmente, lo implementé de una manera en la que lo que me primaba era el desarrollar el componente y probar su funcionamiento lo más rápida y ágilmente posible, para no perderme en optimización demasiado en ese punto, que la experiencia me ha enseñado que hacer optimizaciones en fases tempranas, pueden ralentizar mucho el desarrollo. Sabía que tendría que utilizar NumPy dada la enorme cantidad de cálculos; además es una herramienta la cual quería poner en práctica y aprovechar la coyuntura del proyecto para aprender sobre ella. Al no conocer mucho, implementar los componentes con numpy directamente iba a hacer la curva de dificultad y testeo más pronunciada. Veamos el proceso y resultado.

2.4.7.1. Comparativa de arquitecturas

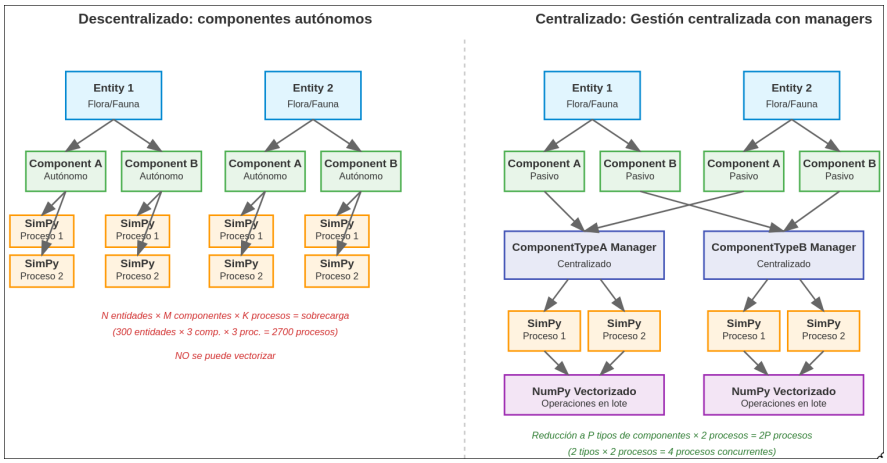


Figura 2.27: Descentralizada vs Centralizada

Versión descentralizada: componentes autónomos

La primera versión fué la versión **descentralizada** y la diseñe de manera que los componentes fueran totalmente autónomos y gestionaran sus propios procesos SimPy para hacer cálculos y a actualizar valores de los parámetros - conceptualmente está muy bien, pero resultaba en un gran número de procesos concurrentes y además, no se podía vectorizar al tener un sistema totalmente descentralizado (realmente, habría formas, pero engorrosas y además un incremento considerable en coste computacional y en memoria ya que habría que hacer búsquedas y organización pesada en cada tick de la simulación).

```python
class PhotosyntheticMetabolismComponent:
    def __init__(self, env, event_notifier):
        self._process = env.process(
            self._update_metabolism())

    def _update_metabolism(self):
        while True:
            energy_change = (
                self._photosynthesis_efficiency *
                self._light_availability -
                self._respiration_rate
            )
            self._energy_handler.modify_energy(energy_change)
            yield self._env.timeout(timer)
```

Figura 2.28: Enfoque descentralizado (concepto, idea): cada componente ejecuta su propio proceso de actualización

Por ejemplo, si tenía 300 entidades con un determinado componente y si cada uno de esos componentes tenían 3 procesos - resultaban 900 procesos SimPy, cada uno haciendo cálculos matemáticos pesados. Esto era claramente un overhead innecesario, no obstante tengo que reconocer que SimPy no dió señales de debilidad en este aspecto, y se mostraba robusto incluso con 2500 procesos concurrentes. Pero, esto es muy probable que no fuera escalable y terminara explotando, así que es una mejor idea la de mantenerlos en un número bajo.

Versión centralizada Entonces, una vez tuve la base de los componentes funcional; al tener tanto cómputo matemático que podría vectorizar con numpy, decidí hacer una refactorización del sistema de componentes e implementar la versión **centralizada**. Para ésta versión, desarrollé sistemas especializados que operan a nivel de tipo de componente (los managers), no de entidad individual. Estos managers son como un registro central; cada vez que se crea un componente - se registra. El manager es que el que lanza y tiene control sobre los procesos de cómputo y actualización y hace cálculos en batches o lotes; con esto aprovecho la vectorización de NumPy.
El componente **sigue teniendo tareas**, como hacer *handling* de ciertos updates, respuestas reactivas a eventos (meteorológicos, estrés)...etc - pero ya no se encarga de estos cálculos.

```python
class PhotosyntheticMetabolismComponent:
    def __init__(self, env, event_notifier):
        self._photosynthesis_efficiency = 0.75
        ComponentRegistry.get_photosynthetic_metabolism_manager().register_component(
            id(self), self)

class PhotosyntheticManager:
    def _update_all_metabolism(self, timer):
        while True:
            components = self._get_active_components()

            respiration_rates = np.array([c.respiration_rate for c in components])
            efficiencies = np.array([c.photosynthesis_efficiency for c in components])
            light = np.array([c.light_availability for c in components])

            energy_changes = efficiencies * light - respiration_rates

            for i, component in enumerate(components):
                component.energy_handler.modify_energy(
                    energy_changes[i])

            yield self._env.timeout(timer)
```

Figura 2.29: Enfoque centralizado (concepto, idea): un solo manager gestiona todos los componentes mediante vectorización. Las **list comprehension**, son algo mejores que loops normales en Python, están implementadas internamente en C y en general son mucho más eficientes (*Zhang et al., 2023*[37])

2.4.7.2. Rendimiento

Comencemos analizando los rasgos generales de cada enfoque:

Métrica	Descentralizado	Centralizado
Procesos SimPy	$O(n)$	$O(1)$
Complejidad algorítmica	$O(n)$	$O(n)$
Eficiencia computacional	Baja	Alta (vectorizada)
Uso de memoria	Alto	Optimizado
Escalabilidad	Limitada	Superior

Tabla 2.2: Comparativa de rendimiento entre ambos enfoques, donde n representa el número de componentes totales en el sistema.

Ahora, veamos unos experimentos que he realizado para medir la diferencia de rendimiento entre ambas implementaciones.

Git commits (SHA):
- **Sin vectorizar:** `376d2c59dac9f45b820171b5fe73780486f2f493`
- **Vectorizando:** `d06280927b12037a9e092c6f0ed40c75e53ee162`

Componentes principales:
Flora: `Growth, Vital, Photosynthetic, Autotrophic/Heterotro. nutr., WeatherAdaptation`
Fauna: `Growth, Vital, Weather`

Mejora calculada con:

$$\text{Speedup} = \frac{T_{\text{sin vec.}}}{T_{\text{vectorizado}}}$$

Nota: Para la actualización del estado, sigue siendo necesario iterar sobre las entidades individuales.

Tabla 2.3: Comparación de rendimiento entre operaciones vectorizadas y no vectorizadas

Mapa Tamaño	Flora	Fauna	Componentes Flora	Componentes Fauna	Sin Vec. (ms)	Vectorizado (ms)	Mejora
125×125	300	300	5	3	230.492,06	160.222,59	43,85 %
125×125	600	500	5	3	436.162,11	293.381,64	48,66 %
125×125	1000	1000	5	3	785.297,13	529.715,67	48,24 %

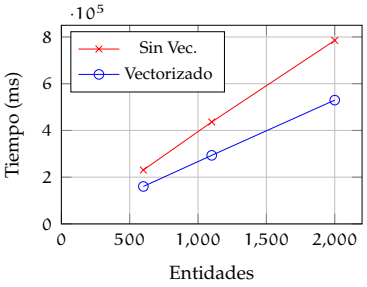


Figura 2.30: Escalabilidad de implementaciones

2.4.7.3. Análisis de rendimiento

Como hemos podido observar, el modelo vectorizado sustituye los n procesos de `SimPy` por un único proceso que opera sobre arrays `NumPy`. Esto elimina el overhead evidente y hace que la gestión de procesos pase de $O(n)$ a $O(1)$. Como inconveniente, mencionar que la complejidad global sigue siendo $O(n)$ porque cada entidad debe actualizar su estado, y esto lo hago en iteraciones normales. Podemos ver también unas mejoras importantes que van desde el 43,8 % al 48,7 % en un mapa de 125 × 125 y hacen uso de hasta 2000 entidades. El *speed-up* medio es 1,47× y la mejora se mantiene al incrementar las cantidades - gracias al paralelismo interno de `NumPy` y a cómo gestiona la caché; ya que el uso de memoria también mejora al almacenar los atributos en bloques contiguos (en contraposición de utilizar objetos Python).

Con lo que, se puede decir que la vectorización ofrece, con diferencia, el mejor rendimiento.

2.5. Métricas ecológicas

Métricas Ecológicas: Sistema de Evaluación

Dado el alto contenido formulaico y que la extensión de la memoria se puede beneficiar de un recorte más, voy a explicar las ideas y exponer las ecuaciones en apéndice. He desarrollado un sistema de **contribuyentes estadísticos** para analizar diferentes áreas del bioma; calculan métricas que luego puedo utilizar para comparar resultados entre simulaciones. Presento las nociones brevemente aquí:

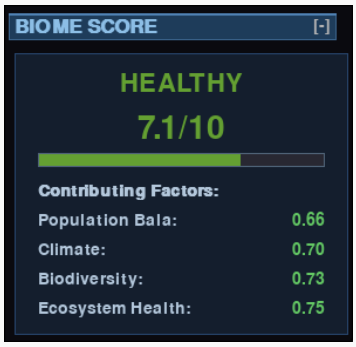


Figura 2.31

Arquitectura de contributors

Equilibrio poblacional: La proporción entre flora y fauna es lo que evalúo aquí. He fijado un ratio ideal de 0.6 flora / 0.4 fauna tras varias pruebas. Tiene penalización asimétrica; el exceso y la escasez de flora no afectan igual - el exceso de flora no lo penalizo tan severamente como la escasez (nos puede llevar a colapso trófico).

Adaptación climática: Me centro en medir si el clima actual es bueno para cada tipo de bioma; me he basado en la noción que documenta (*Whittaker, 1975*[38]) en la que habla de clasificación de biomas basado en variables climáticas. Hago algo parecido, tengo en cuenta los factores medioambientales y con ellos penalizo las desviaciones con respecto a lo deseado para cada bioma.

Biodiversidad: Utilizo dos enfoques a la vez - el índice de **Shannon-Wiener** que nos da una idea de la distribución de especies, y un análisis de relaciones depredador-presa (aquí me baso en las pirámides tróficas de (*Lindeman, 1942*[39])). Al final los combino con una media ponderada - la idea es detectar patrones de dominancia y representarlo en el índice (si bajo, es inestable).

Salud ecosistémica: Incluyo varios factores, pero agregados (medias): estrés, tamaño, toxicidad, etc. Hacen la labor de una especie de **bioindicadores**; me he basado en intuición.

Clasificación

Para la puntuación final, lo que hago es combinar todos contributors de manera ponderada. Al final, todo esto es para clasificar la calidad ecosistétmica en cinco categorías: ***Crítico, Inestable, Moderado, Saludable, Edén***, así tenemos una noción interpretable de su estado.

Para especificaciones y fórmulas detalladas, consultar Apéndice: ***Métricas Ecológicas*** *(Apéndice, p. 117)*

2.6. Algoritmo de votación distribuida de dormancia en flora

Algunas plantas entran en dormancia para sobrevivir situaciones difíciles: reducen drásticamente su metabolismo, consumen lo mínimo necesario, disminuyen su tasa de respiración hasta un 75 %, ralentizan otros procesos fisiológicos, etc. He implementado un algoritmo de `polling` distribuido de componentes y sincronización para gestionar el estado de dormancia de la flora y que sea coherente. Puede verse en el **Algoritmo de dormancia** (Apéndice, p. 132)

Bioma vivo - latido digital

Para terminar, démosle vida al bioma y veamos cómo trabajan los componentes de manera conjunta. He lanzado varios experimentos con una única entidad de flora en diferentes condiciones ambientales. Con ello intento comparar el comportamiento a nivel biológico de la especie, con parámetros optimizados para biomas específicos versus especies fuera de su hábitat natural. La simulación ha sido de 50.000 eventos, 5.000 por era, lo que se traduce en 69.4 años.

Para la **configuración experimental** he utilizado dos especies con características muy diferentes - el **oak tree** lo he configurado para que sea óptimo en clima tropical, simplemente para las pruebas:

Parámetro	Oak Tree	Arctic Willow
Temperatura óptima (°C)	28.0	-10.0
Resistencia al frío	0.0	0.95
Resistencia al calor	0.0	0.20
Tamaño máximo (m)	3.0	1.0
Eficiencia fotosintética	0.70	0.50
Tasa de respiración	0.35	0.10
Actividad metabólica	0.80	0.40
Esperanza de vida (años)	40	17

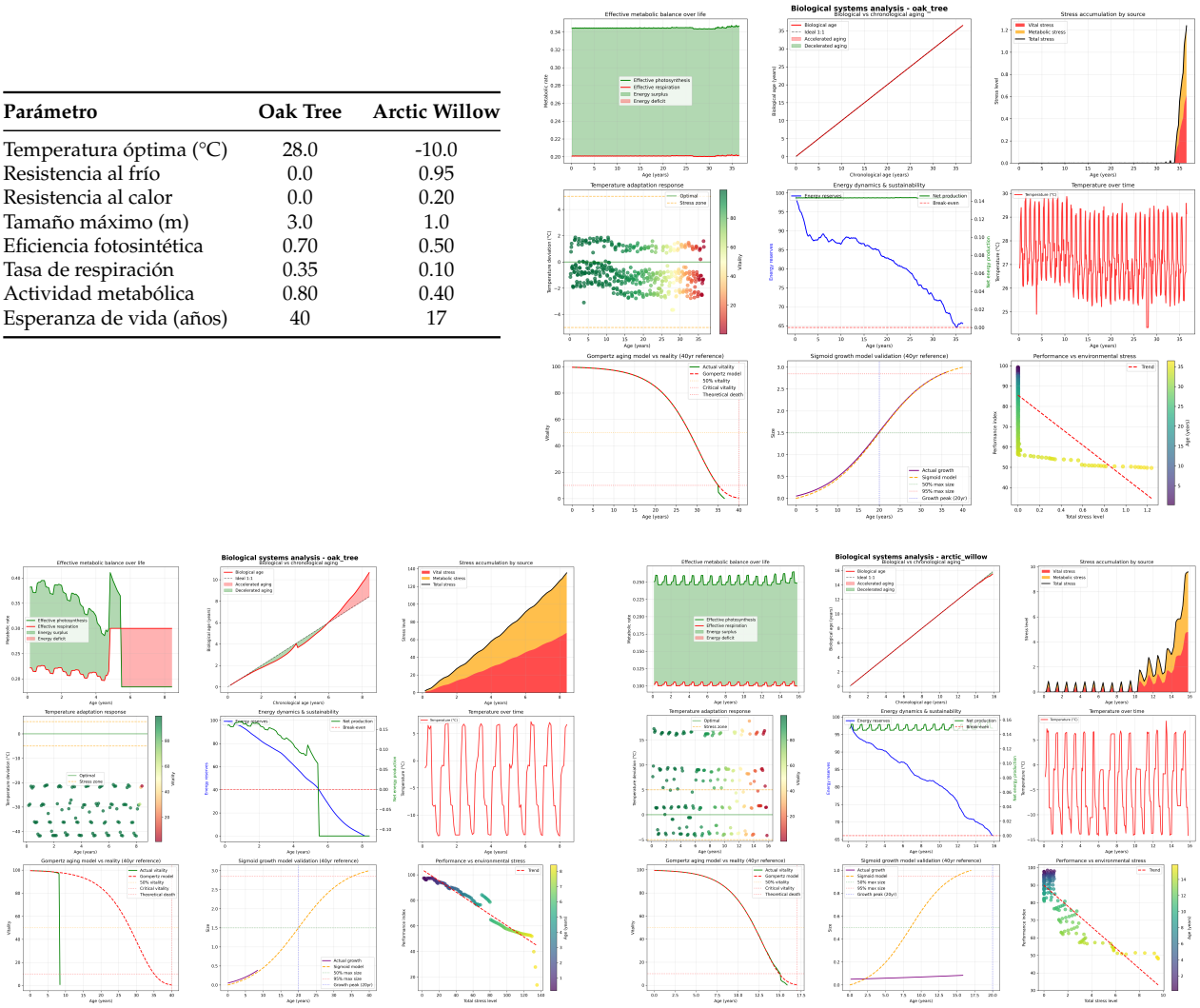


Figura 2.35: Arriba derecha: **Oak tree - tropical** | abajo izda: **oak tree - tundra** | abajo dcha: **arctic willow - tundra**

En la gráfica de *Energy Dynamics & sustainability*, la linea roja, es el **break even**, esto es: $\texttt{net\_energy\_production} = \texttt{effective\_photosynthesis} - \texttt{effective\_respiration}$, lo que implica que si por encima de la linea la entidad produce más energía de la que consume, si por debajo, consume más de lo que produce. En la gráfica de Stress accumulation by source, estamos viendo el estrés que la flora sufre; incluye el estrés térmico que cada componente de penalización recibe. Y, para terminar, en Performance vs enviornmetnal stress, la performance es una ponderación tal que $\text{performance} = \left(0{,}4\,\text{vitality\_ratio} + 0{,}3\,\text{energy\_ratio} + 0{,}3\,\text{effective\_photosynthesis}\right) \times 100$.

En el primer experimento tenemos un oak tree (roble) con temperatura óptima de 28°C en un bioma tropical (Imagen 1 – arriba derecha). Como era de esperar, mantiene un balance metabólico estable durante toda su vida útil, con fotosíntesis efectiva superior a la respiración, lo cual es muy positivo (superávit estable (zona verde)). El modelo de envejecimiento de **Gompertz** - es casi ideal (al final hay un pequeño cambio); la tasa de envejecimiento biológico está estable en 1.0 prácticamente toda la vida (gráfica fila 0, columna 1).

La acumulación de estrés se mantiene mínima hasta el final - aquí los procesos naturales de senescencia ya empiezan a predominar. En general, un desarrollo muy bueno.

En el segundo experimento, traslado la misma especie a un bioma de **Tundra** (Imagen 2 – abajo izquierda), y podemos ver el **impacto severo del desajuste térmico**. Aquí ya sufre estrés continuo por las temperaturas que oscilan entre -15°C y 5°C, muy por debajo de su óptimo de 28°C, como se puede ver en la gráfica de Temperature adaptation response y en la de Stress accumulation by source.

Aquí comienza lo interesante – la **tasa de envejecimiento biológico se acelera dramáticamente**; esto, efectivamente, reduce la longevidad proyectada y la entidad **termina muriendo mucho antes** del lifespan de su especie (40 años). El balance energético se vuelve insostenible poco antes del año 6; las reservas se agotan y la entidad no puede mantener sus funciones vitales.

Pequeño inciso – podemos observar que el estrés térmico reduce la eficiencia fotosintética y llega un punto donde incrementa mucho la tasa de respiración - esto lleva a déficit energético; y a su vez esto acelera el envejecimiento y aumenta el estrés metabólico y se traduce en una caída de performance temprana si comparamos con el experimento anterior. La idea era esta – un bucle de retroalimentación, que en este caso es negativo y termina siendo letal. Se ve claramente que el factor que limita a la flora es la inadaptación climática – aquí es donde más adelante, entrará el factor de evolución.

Por último, y para contrastar, veamos el mismo entorno *castigador* del bioma **Tundra**, pero con una **especie nativa: Arctic Willow** (sauce ártico) (Imagen 3 – abajo derecha); está diseñada para condiciones de frío. Tiene resistencia al frío de 0.95 y temperatura óptima de -10°C, y mantiene estabilidad metabólica durante sus 17 años de lifespan.

La tasa de envejecimiento es estable cerca de 1.0 durante toda la simulación, salvo hacia el final que podemos ver una pequeña deceleración (lo cual hace pueda vivir un poco más que el lifespan proyectado). El balance energético está equilibrado (predominio de zona verde), y la adaptación térmica – el 0 es el óptimo (-10ºC) y podemos ver cómo los puntos están dispersos por la desviación, esto implica que ésta entidad tiene resistencia al frio, pero hacia el otro extremo (el calor) no tiene tanta resistencia – aguanta mucho mejor que el oak tree, y el frio lo soporta muy bien, está en su zona de confort, pero aún le faltaría por adaptarse mejor a temperaturas más altas. Con la desviación podemos observar el **efecto explicado con respecto a la adaptación gaussiana** – el -5 implica que es capaz de soportar temperaturas inferiores de hasta -15º, pero el 15 implica que 5º, es demasiado para esta planta. Recordemos que ésta amplitud se regula mediante σ y las resistencias al frío/calor.

Con esto, podemos determinar que especies bien adaptadas prosperan en su entorno natural, pero las especies desplazadas sufren en climas que no son los suyos, lo que intenta representar la realidad biológica.

CAPÍTULO 3

Sistema multi-agente

3.1. Agentes

Uno de los pilares fundamentales de la plataforma es el sistema de agentes. Me he basado en el concepto clásico de agentes en Inteligencia Artificial: entidades autónomas que perciben su entorno, toman decisiones con esa información y ejecutan acciones que modifican dicho entorno. Todos los agentes existen dentro del Bioma y siguen un ciclo común gracias a una una interfaz unificada, sin importar sus tareas específicas o modelos subyacentes:

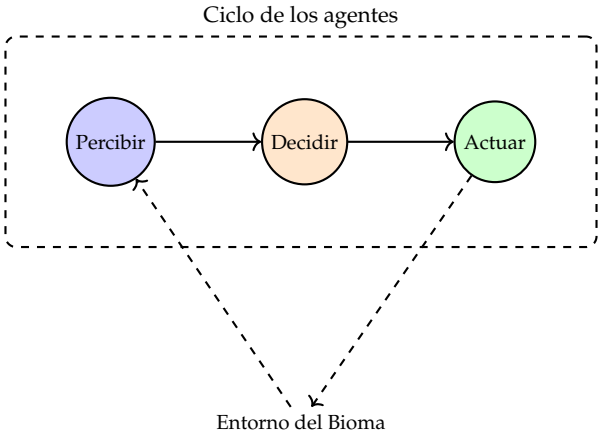
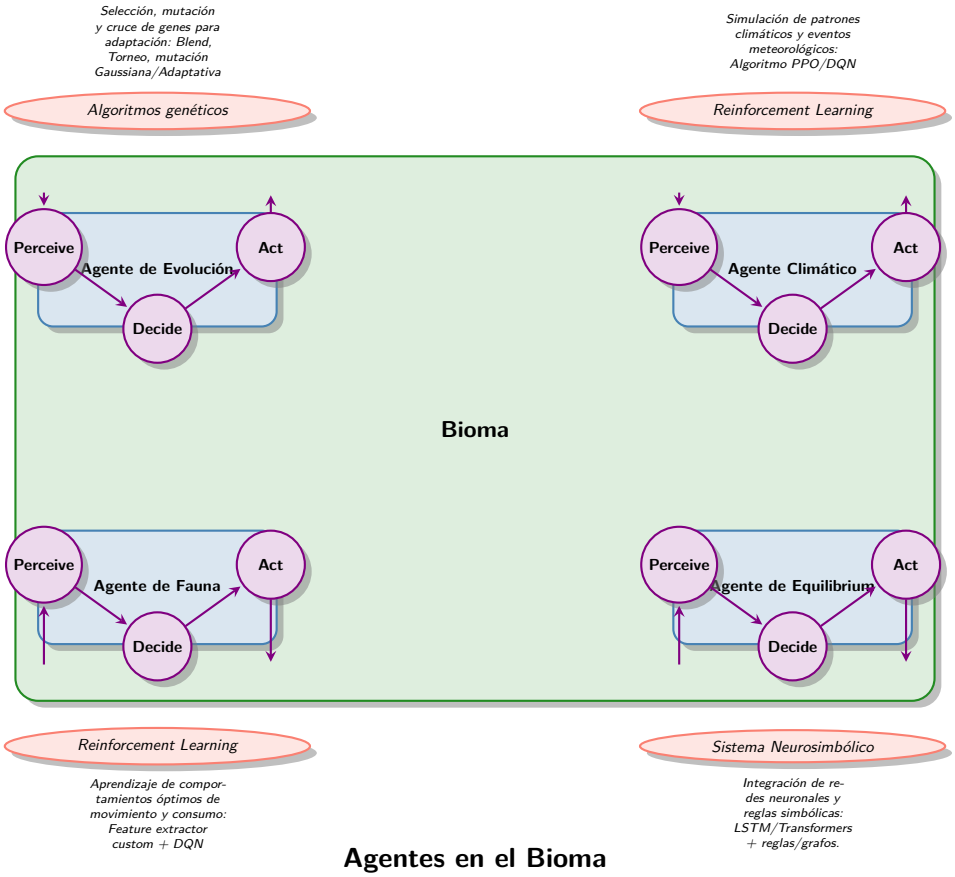


Figura 3.1: Interfaz común a **todos** los agentes

He creado cuatro tipos de agentes - cada uno cumplirá sus funciones específicas dentro del ecosistema y los iremos viendo en este orden:

- **Agente de evolución**: Cada especie de flora y fauna posee ciclos de selección natural y adaptación genética (por especie)
- **Agente climático**: Dirige las condiciones atmosféricas y eventos meteorológicos.
- **Agente de fauna**: *Cerebro* o comportamiento de las entidades de fauna.
- **Agente de Equilibrium**: Se centra en intentar mantener homeostasis/balance en el bioma.

Todos los agentes disponen de un modelo que ha de ser preentrenado, salvo el agente de evolución que se basa en algoritmos genéticos.



Agentes en el Bioma

Como ya he explicado, los agentes funcionan de manera autónoma pero colaboran indirectamente (a veces directamente) al interactuar con un entorno compartido - el bioma - de modo que las acciones de cada uno tienen un impacto directo y generan cambios que los otros agentes perciben. Mi idea desde el principio era que surgieran sinergias complejas entre ellos para poder estudiar comportamientos complejos - pero balancear un sistema holístico tan grande no es nada trivial - no obstante, para el tiempo que he tenido para planificar, diseñar y desarrollar el proyecto, creo que los resultados son decentes y, en general, estoy satisfecho con lo conseguido, he aprendido mucho.

3.2. Agente de evolución

3.2.1. Introducción

La evolución en la naturaleza sucede mediante selección natural y aplica variaciones genéticas a lo largo de generaciones. He intentado modelar esto mismo en el proyecto, aunque de forma muy modesta.

Desde el principio quería basarme en **algoritmos genéticos** - siempre me han llamado la atención y llevaba años queriendo ponerlos en práctica. Investigué alternativas como NEAT (NeuroEvolution of Augmenting Topologies), que me pareció fascinante en lo poco que llegué a ver. Sin embargo, por la fase de desarrollo en que me encontraba y la carga de trabajo aún pendiente, elegí los algoritmos genéticos: más intuitivos y rápidos de prototipar gracias a frameworks con buen soporte. El nivel de complejidad que yo requiero no es alto, al ser una versión bastante relajada. Como nota, NEAT queda pendiente para futuras mejoras.
Para desarrollar el modelo genético me he apoyado en la biblioteca **DEAP** (Distributed Evolutionary Algorithms in Python). Me ofrece un toolbox completo y además, me ha servido para familiarizarme con una de las herramientas más extendidas para AGs.

3.2.2. A medio camino entre $(\mu + \lambda)$-EA y EA con generaciones solapadas

El modelo evolutivo que he plasmado para los biomas, difiere en varios aspectos de los algoritmos genéticos estándar que he visto se suelen utilizar:

- Aquí, la evolución es un **proceso que ocurre en una simulación continua**, en lugar de ciclo iterativo hasta convergencia. Se distribuye durante toda la ejecución.
- Dejo que los individuos desaparezcan por **causas naturales** (vejez, escasez de recursos, depredación); no he hecho descarte explícito.
- La evolución ocurre en **ciclos acelerados**: cada evento reproductivo genera nuevos individuos con genes cruzados y mutados.
- Existe **presión selectiva implícita** - sólo quienes sobreviven al entorno llegan a *'reproducirse'*.
- He integrado la capacidad reproductiva en el propio agente evolutivo, ya que aún no existe un sistema de ciclo reproductivo separado. Por eso los intervalos de ejecución son cortos, se verá más adelante.

Antes del desarrollo, intenté buscar referencias que justificaran mi planteamiento. Encontré conceptos como **Evolving Populations with Overlapping Generations** (*Rogers y Prügel-Bennett, 2000*[40]), que se acerca al comportamiento natural, pero mi principal inspiración fue **A. E. Eiben**, tanto su libro sobre algoritmos de estado estable (*Eiben y Smith, 2015*[41]) como otras publicaciones suyas. He adoptado ideas de su taxonomía de control de parámetros (*Eiben, Hinterding et al., 1999*[42]); ya que en mi proyecto el **número de descendientes** λ **se adapta de forma dinámica**, depende del entorno y la esperanza de vida de su especie. Por lo que he explicado previamente, mi idea es **incorporar los descendientes a la población tan pronto como nacen**, algo similar a lo que mencionan Eiben y Smith, 2015 sobre los Steady-State GA; pero en estos la población se actualiza de manera parcial y continua. No obstante, mi caso no es un Steady-State clásico, ya que este requiere **tamaño constante de población** y elimina un individuo por cada nuevo descendiente.

Con esto, obtenemos un modelo que ejecuta ciclos e inyecta individuos con genes evolucionados en cada ejecución. Aunque llamo al algoritmo genético con pocas generaciones (`ngens`) en cada ejecución, consigo algo parecido a la vertiente $\mu + \lambda$, porque al ser continuo, algunos individuos mueren por causas naturales y otros no; pero los que llegan al siguiente ciclo evolutivo son tanto padres como individuos con genes evolucionados. No sigo estrictamente la vertiente $\mu + \lambda$, solo me inspiro en ella. Además, se solapan generaciones y acelera el ritmo evolutivo, he investigando sobre esto he descubierto que se alcanza el equilibrio en la mitad de eventos reproductivos (*Rogers y Prügel-Bennett, 2000*[43]) justo por ello, lo que viene bien para nuestra simulación acelerada.

Si nos fijamos en la figura 3.2, cada especie ejecuta pequeños ciclos de evolución de forma separada. Cuando se dispara un evento de reproducción, selecciono padres, aplico *crossover* y mutación, traduzco genes a componentes e *inyecto* los nuevos individuos dentro de la simulación. **Sin detención, sin poblaciones discretas, siempre en flujo**. Mi enfoque ha sido el de intentar que simule lo que ocurre en un bioma real, donde la vida y la selección natural nunca se detienen. Considero que gracias a éste método **se refuerza la capacidad de *exploración* del espacio de soluciones**; al ensayar múltiples ciclos sin que afecte a la población real. Pero además, al reinyectar solo los *k_best*, el sistema *explota* las regiones de mejor calidad o fitness.

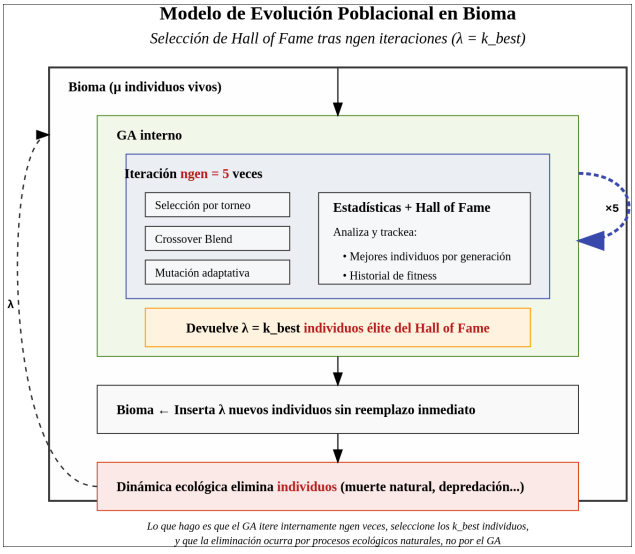


Figura 3.2: Modelo evolutivo

3.2.3. Arquitectura del agente evolutivo

3.2.3.1. Registro de agentes evolutivos

Existe un agente evolutivo **por especie**, a diferencia de otros agentes que son únicos para todo el sistema. Mi intención con esto es que cada especie evolucione según su propio ritmo y condiciones. Para gestionar esto, existe un registro de agentes evolutivos donde se centralizan. A modo de ejemplo podemos ver la siguiente imagen - cada agente dispone de sus propios tiempos, parámetros, etc:

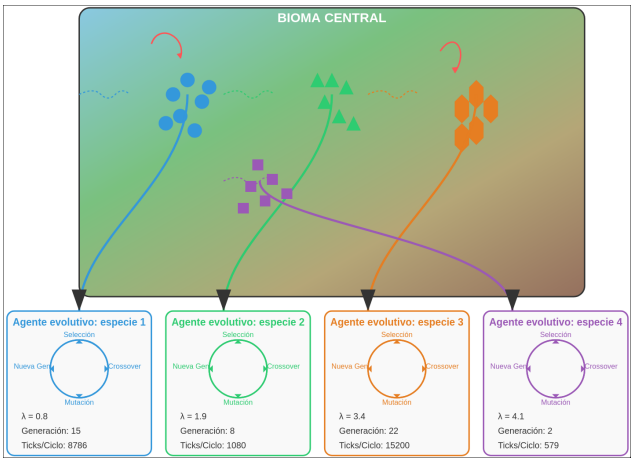


Figura 3.3: Agente por especie

1 → 2 → 3

Percepción: Se observan entidades vivas de la especie, se recopilan datos climáticos recientes y estadísticas poblacionales.

Decisión: Para decidir, calculo el kbest, se evalua fitness; selección por torneo, y se ejecuta cruce y mutación.

Acción: Se forjan nuevas entidades con los genes evolucionados. También registro datos, y ajusto parámetros del ciclo evolutivo...etc

3.2.3.2. Ciclos evolutivos

Cada agente tiene un *timer* para su ciclo, pero al principio está basado en el `lifespan` de la especie; éste se va ajustando dinámicamente a lo largo de la simulación. Con esto, consigo los siguientes efectos:

- **Especies de vida corta - evolución más rápida**, ciclos más frecuentes.
- **Especies longevas - evolución más gradual**, ciclos más espaciados.
- Ajuste dinámico; el intervalo entre generaciones va variando según condiciones (emergencias, ajustes externos de la plataforma (como se verá más adelante), etc).

Ya que no existe un ciclo reproductivo implementado, el agente de evolución hará esta labor de manera totalmente implícita. El ciclo para la **generación 0** lo inicializo tal que:

$$c_{evol,0} = 0{,}05 \cdot L_{span} \tag{3.1}$$

Y durante la simulación, este valor se ajusta tras cada ciclo evolutivo, si $c_{evol} < 0{,}4\,\bar{L}_{actual}$, se incrementa según:

$$c_{evol,n+1} = c_{evol,n} + \bar{L}_{actual} \cdot \epsilon \tag{3.2}$$

donde $\epsilon \in [0{,}001, 0{,}005]$ es un factor aleatorio y $\bar{L}_{\text{actual}}$ el **lifespan medio de la generación actual** (ya que la esperanza de vida, irá cambiando por individuo). Con ello me aseguro que el ciclo no sea ni demasiado corto (para evitar generaciones con poco tiempo para adaptarse) ni demasiado largo.

3.2.4. Representación de genotipos

Los genotipos son el conjunto de genes - he definido genotipos diferentes para la flora y la fauna, **cada gen será un factor asociado a un componente**. De esta manera puedo traducir y hacer transformaciones bidirecionales entre genes y componentes. Considero que encaja bastante; al modelarlo de esta manera represento las características evolutivas de cada organismo o proceso biológico de la entidad.

Genotipo de flora - especies vegetales

Gen	Rango	Gen	Rango
growth_modifier	[0,5, 2,0]	growth_efficiency	[0,3, 1,0]
max_size	[1,0, 5,0]	max_vitality	[50,0, 200,0]
aging_rate	[0,1, 2,0]	health_modifier	[0,5, 1,5]
base_photosynthesis_effici	[0,3, 0,7]	base_respiration_rate	[0,08, 0,25]
metabolic_activity	[0,3, 1,2]	max_energy_reserves	[50,0, 200,0]
cold_resistance	[0,0, 0,9]	heat_resistance	[0,0, 0,9]
optimal_temperature	[−10,0, 40,0]	nutrient_absorption_rate	[0,1, 0,9]
mycorrhizal_rate	[0,01, 0,2]	base_nutritive_value	[0,3, 1,0]
base_toxicity	[0,0, 0,8]		

Genotipo de fauna - especies animales

Gen	Rango	Gen	Rango
growth_modifier	[0,5, 2,0]	growth_efficiency	[0,3, 1,0]
max_size	[1,0, 5,0]	max_vitality	[50,0, 200,0]
aging_rate	[0,1, 2,0]	health_modifier	[0,5, 1,5]
cold_resistance	[0,0, 0,9]	heat_resistance	[0,0, 0,9]
optimal_temperature	[−30,0, 50,0]	hunger_rate	[0,8, 1,5]
thirst_rate	[1,2, 2,0]	metabolism_efficiency	[0,5, 1,2]

3.2.4.1. Codificación/decodificación: genes ↔ componentes

Lo bonito es que esto me permite hacer una adaptación independiente de cada tipo de organismo, y crear una equivalencia *materializable* entre genes y componentes, para poder introducir individuos evolucionados *recién nacidos* en la población de su especie. El proceso, de manera simplificada, es:

1. Se extraen los genes de las entidades actuales en la población.
2. Estos genes se modifican mediante operadores genéticos.
3. Una vez modificados, se convierten en componentes para nuevas entidades o individuos.

Como he comentado previamente, se crea un ciclo completo; la información genética fluye desde las entidades actuales hacia nuevas generaciones gracias a el proceso evolutivo que lo va transmitiendo.

3.2.5. Fases del algoritmo evolutivo

3.2.5.1. Fase de percepción

Gracias a los sistemas de recolección de datos y a los mecanismos para exponer esa funcionalidad a otras partes del framework, ahora podemos obtener la información que cada agente necesita de manera muy sencilla. El agente recopila información con objeto de evaluar y modificar la población:

Observaciones

- **Datos de población**: Entidades vivas de su especie. Extrae sus genes
- **Datos climáticos**: Recopila información sobre condiciones climáticas recientes. Pero con un twist:

Sobre la media móvil ponderada exponencial (EWMA)

> Voy a hacer un pequeño inciso aquí para comentar algo que he disfrutado mucho de por fin poder aplicar. En la asignatura de AARN nos introdujeron al concepto de media móvil ponderada exponencial (en el contexto de SGD con Momentum, de **Ilias Sutskever**). Me encantó la idea de usar conceptos físicos como la inercia para mejorar el entrenamiento. No pude aplicarlo en su momento pero, me pareció muy útil tanto para reducir el ruido de los datos como para trazar una **tendencia con memoria** pero - y he aquí lo que me hizo querer aplicarlo en este contexto - además dando especial **importancia a los eventos más recientes**.

Con ese pequeño inciso, se puede ver cómo EWMA resulta apropiada para modelar implícitamente cómo la **evolución natural hace que las especies se adapten a cambios más recientes, pero sin perder memoria de condiciones pasadas**. Si una especie evoluciona, debe considerar toda su historia, pero su adaptación tiene que estar muy enfocada en el estado reciente del entorno. Por ejemplo, si el clima de una zona ha cambiado gradualmente con el tiempo, pero en los últimos 100 años hace mucho más frío, las especies deben recordar la trayectoria completa, pero sobre todo aclimatarse a ese frío reciente.
Veamos brevemente cual es el proceso que he implementado para obtener estos datos. Se divide en tres etapas:

Etapa 1: Recolección y procesamiento

1. **Recolección diaria**

   Cada día registro la temperatura, humedad, precipitación, presión atmosférica y estación actual.

   Los datos se organizan en un dataframe con:

   - Ciclo evolutivo
   - Día y mes
   - Mediciones climáticas

2. **Promedios mensuales**

   Al final de cada mes hago una agregación - calculo las medias mensuales de todas las variables.

   Me ayuda a comparar periodos y además reduzco el ruido de fluctuaciones diarias.

3. **Media móvil ponderada**

   Sobre esos promedios mensuales aplico una EWMA con span $= 6$.

   Con este span, estoy diciendo que otorgue más peso a los datos recientes - pero que no ignore el pasado:

   - Span de 6: $\alpha = \frac{2}{6+1} \approx 0{,}2857$
   - Formula:

$$\text{EMA}_t(V) = \alpha \cdot V_t + (1-\alpha) \cdot \text{EMA}_{t-1}(V)$$

Esto implica que cada nueva observación aporta un 28.57 % al valor suavizado, y el resto del EWMA

previo *se acuerda* de un 71.43 %: me ayuda a **enfatizar tendencias y filtrar ruidos**.

Dataframes de datos climáticos

- 1. Recolección diaria de métricas
- 2. Agregación mensual
- 3. Cálculo de EWMAs por variable
- 4. Indexación por ciclo evolutivo

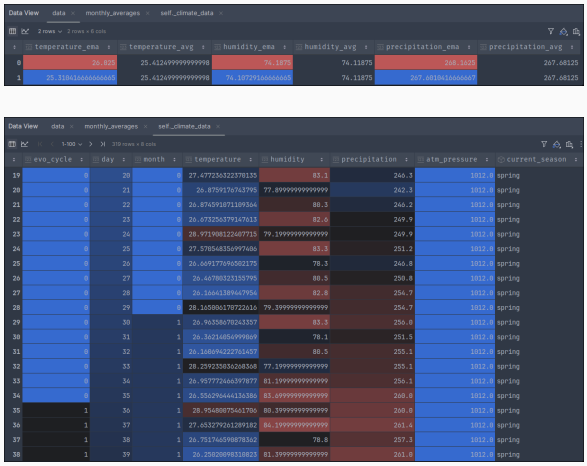
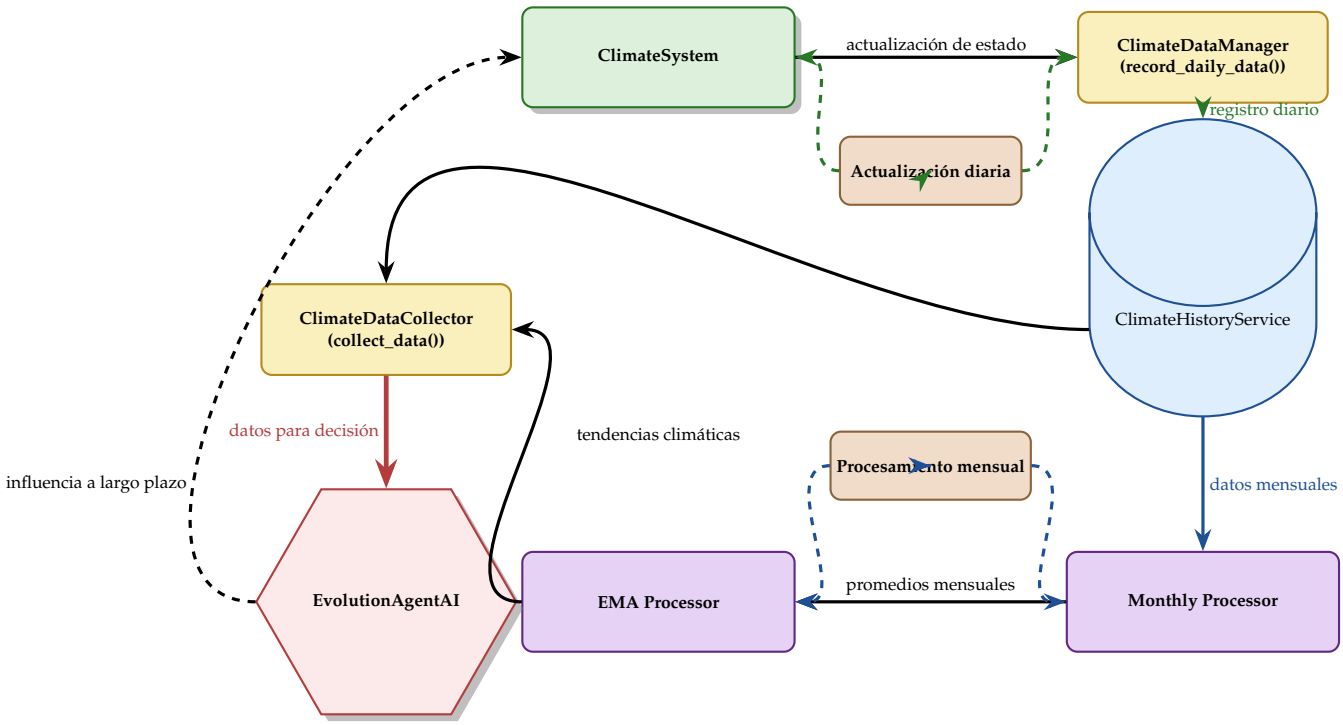


Figura 3.4: Dataframes de agregados (arriba, EWMA + medias) e historial climático durante simulación (debugger del IDE + plugin para dataframes)



Recapitulando, la observación que el agente hará contendrá información sobre la población de la especie que corresponda, y datos climáticos procesados.

3.2.5.2. Fase de decisión

Vamos a desgranar el proceso para obtener nuevos individuos con genes evolucionados que se muestra en la figura 3.2.

Como mencioné anteriormente, vamos a introducir un número de individuos nuevos en la población. Para esto decidí usar un método con **elitismo**. Recordemos que realizo una evolución de la población actual de forma meramente virtual, con un número determinado de generaciones. Escogí 5 generaciones porque con valores mayores converge demasiado rápido y llega a meseta muy temprano. Por otro lado, si uso solo 1 generación tiene valor prácticamente nulo para la versión acelerada que busco.

Entonces, una vez evolucionadas estas 5 generaciones, me quedo con **los mejores individuos y sus genes**, sin importar de qué generación vengan (**hall of fame**). Para determinar este número (`k_best`) lo hago de forma dinámica:

Cálculo del k-best

- Tamaño actual de la población
- Esperanza de vida media
- Amplificación por tendencias poblacionales recientes

Modelo matemático:

$$k_{base} = \text{máx}\{3, \text{mín}\{10, 0{,}2\,N\}\} \quad (1)$$

$$\phi = 1 + 0{,}1\,\frac{\bar{L} - L_0}{L_0} \quad (2)$$

$$\lambda = k_{best} = k_{base}\,\phi\,A \quad (3)$$

Símbolo	Descripción
N	Tamaño de la población.
$\bar{L}$	Lifespan medio.
L_0	Lifespan base de especie.
A	Amplificación por tendencias.
ϕ	Factor de longevidad.
k_{base}	Tamaño base del conjunto candidato.
k_{best}	Número final seleccionado.
λ	Alias para k_{best} por símil con estándars.

Escalado de k_{base}
La idea es que el factor k_{base} escala con el 20 % (0,2) del tamaño de la población N y lo limito entre 3 y 10. Así me aseguro que poblaciones más grandes tengan mayor representación, pero con límites ajustados a mis pruebas. Se puede desacotar completamente o ajustar según la simulación que vayamos a hacer.

Con esto determinamos automáticamente el número de individuos a introducir según si su especie vive más o menos de lo esperado y según el tamaño poblacional.

3.2.5.3. Evaluación del fitness

Al igual que con las métricas ecológicas, con objeto de que reducir la carga de contenido formulaico y además, recudir la extensión del documento, expongo los conceptos que he seguido a la hora de diseñar la función de fitness. *Para detalles sobre las ecuaciones, consultar Apéndice:* ***Evolución - fitness de entidades*** *(Apéndice, p. 119)*

Para evaluar la calidad de los individuos y cuan viables sus fenotipos son, he definido una **función de evaluación multicomponente**, para poder ir contrastando la capacidad de supervivencia en diferentes contextos climáticos. Fauna y flora comparten muchas cosas, pero hago diferencia explícita.

Marco conceptual

Adaptación térmica: Una vez más, distribuciones gaussianas - centradas en la temperatura óptima de cada especie. La **resistencia al frío/calor** determina la amplitud de tolerancia dependiendo de lo que nos digan los datos - e.g: si el EMA es una temperatura baja, se contrastará la resistencia al frio. Con esto, una desviación mayor pensalizará más.

Balance energético: Para flora, evalúo la **viabilidad fotosintética**; lo importante es el ratio fotosíntesis/respiración efectivas, y tener en cuenta factores ambientales (luz, temperatura, agua). He implementado **constraints biológicos** que penalizan configuraciones inviables. Por otro lado, para fauna, quiero optimizar las **tasas metabólicas** (hambre/sed) para buscar equilibrio entre eficiencia energética y supervivencia.

Especialización vs. generalización: También existen **trade-offs evolutivos**; especializaciones extremas en un rasgo penalizan otros, pero esto lo he hecho de manera acotada - esto es, que la eficiencia fotosintética suba y el respiration rate baje, por ejemplo, pero con cuidado para que el algoritmo no lo abuse.

Fitness según tipo de organismo

Flora – enfoque autótrofo: Priorizo eficiencia fotosintética, gestión hídrica, resistencia a estrés etc. Lo que se evalúa es la capacidad de **absorción de nutrientes** y **tolerancia ambiental**.

Fauna – enfoque heterótrofo: Me centro en **optimización metabólica**; gestión de reservas energéticas (dependiendo de su rol, si es depredador necesita más energia). También tengo en cuenta **umbrales de supervivencia** mínimos, trade-offs entre tamaño físico y eficiencia térmica...etc.

Validaciones y restricciones

He implementado **múltiples capas de validación**: desde constraints físicos básicos hasta penalizaciones por configuraciones un poco irrealistas (extremos, etc). Mi idea en todo momento ha sido la de intentar que se explore el espacio fenotípico que tenemos, pero intentar dirigir la evolución hacia soluciones óptimas. No obstante, **creo que he granularizado demasiado**, es algo similar al sistema de rewards de Reinforcement; es fácil dejarse llevar e intentar abarcar todos los casos. Estoy convencido de que podría simplificarse sobremanera.

3.2.5.4. Selección - Torneo

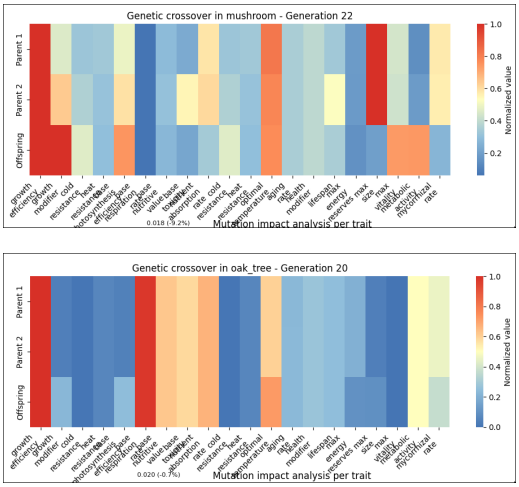
Desde el principio tenía la mirada en el método elitista de coger los k_best individuos, lo que me permite explotar las soluciones - pero como nos han enseñado en la carrera en multitud de asignaturas, siempre hay que encontrar ese *punto dulce* entre exploración y explotación.
Por eso me decanté por la selección por torneo - aunque al final terminaré quedándome con los `k_best`, quiero que durante el proceso **haya al menos algo de exploración**. Para seleccionar individuos a reproducirse, busco un número de participantes que me de menor presión selectiva

por dos motivos: **1)** a más presión, más riesgo de quedarnos atascados en óptimos locales (más probabilidad de que *toquen* individuos aptos) **2)** con menor presión, exploramos más el espacio de soluciones - más diversidad genética.

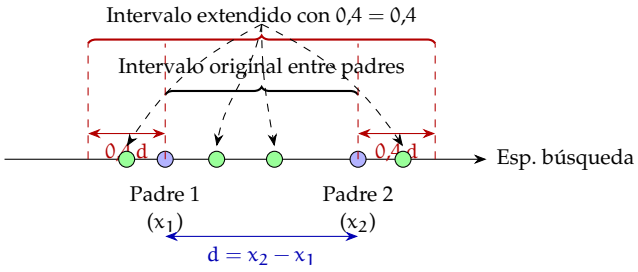
Valor final: 2 individuos para cada torneo, ya que con k=1 es puramente aleatorio dentro de nuestro espacio de búsqueda (gana automáticamente el que sea escogido).

3.2.5.5. Cruce genético - Crossover blend

Para éste paso de todo Algoritmo genético y de nuevo, con espíritu explorador, he elegido el Crossover Blend (BLX-α) ya que es un método de cruce que nos permite extender la búsqueda más allá de los rangos parentales. Además es un método muy adecuado para variables continuas - que, en mi caso, lo son.



Crossover de mushroom/oak: evolución generacional (ampliar)



Representación del operador BLX-α (Blend crossover)

Crossover genético en hongo y roble mediante Blend-α (con α=0.4).
Se puede ver cómo algunos rasgos como `photosynthesis_efficiency` y `metabolic_activity` (hongo) mejoran en el hijo gracias al cruce (rojo = valor alto, azul = valor bajo) Lo bueno de BLX-α es que explora más allá de los valores intermedios entre padres, y con esto balanceamos exploración del espacio genético y explotación de features.
Nota: Me ha llamado la atención el cruce de los dos padres *rojos* idénticos, donde el hijo es diferente. No he conseguido identificarlo a ciencia cierta, pero estoy casi convencido de que es debido a una mutación, ya que primero se cruza y luego se muta.

Parámetros finales elegidos:

- $\alpha = 0{,}4$: extiende en un 40 % el intervalo entre padres y esto me permite explorar zonas adyacentes sin dispersar la búsqueda.
- `cxpb` $= 0{,}66$: con este valor consigo que en dos de cada tres cruces, se mezclen genes posiblemente buenos y se acelere la convergencia.
- `mutpb` $= 0{,}30$: una mutación cada tres descendientes (recordando que estamos en evolución acelerada) - me parece un buen ratio para reinyectar diversidad en la población original y salir de óptimos locales ó, como bien dijo el profesor de la segunda parte de la asignatura de Operations Research:

"Si la pelota se queda atascada, le pegamos una patada y que busque en otro lado"

Figura 3.5: Cruce genético - Crossover blend (BLX-α): evolución y parámetros

3.2.5.6. Mutación

Si le hemos pegado una patada a la pelota, no es trivial el determinar donde caerá. Para modelar esa incertidumbre he implementado un *operador de mutación* con dos modos compatibles con DEAP En un primer momento, me percaté de que algunos genes tenían problemas al evolucionar, ya que al operar la mutación sobre sus valores normalizados, era posible que un cambio pequeño para un gen fuera enorme para otro.

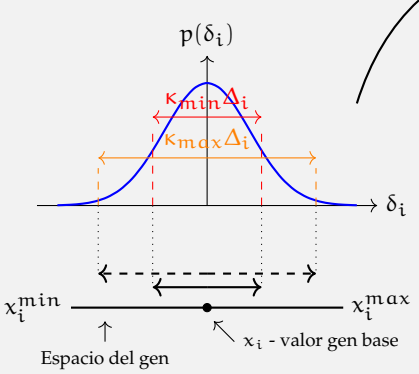
Entonces implementé una función customizada para que durante el proceso se desnormalizara y se aplicara el cambio sobre el valor en su rango natural, para después volver a normalizar y continuar el proceso evolutivo. Mejoró mucho, pero no me terminaba de gustar, con lo que terminé haciendo un operador de mutación con dos funcionalidades diferentes y preparado para ser integrado con DEAP sobreescribiendo el método `__call__`:

Operadores de mutación

1. Mutación adaptativa.
A cada gen x_i se le aplica un desplazamiento porcentual acotado (en su espacio desnormalizado):

$$x_i' = \text{clip}(x_i + \alpha_i\, x_i,\ [x_i^{\text{mín}}, x_i^{\text{máx}}]), \quad \alpha_i \in [-p_{\text{máx}}(L), p_{\text{máx}}(L)].$$

Al principio ajustaba $p_{\text{máx}}(L)$ en función del *lifespan* de la especie, con la idea de que especies longevas toleraran mutaciones más grandes. Calculaba un ratio basado en el *lifespan*. Al final pasé a la gaussiana custom, y dejé esta opción con un máximo del 20 % de cambio sobre el rango natural de cada gen. Este es el truco para preservar la escala y que no haya saltos grandes.



2. Mutación gaussiana.
A cada gen x_i se le aplica un desplazamiento δ_i muestreado de una normal:

$$x_i' = \text{clip}(x_i + \delta_i,\ [x_i^{\text{mín}}, x_i^{\text{máx}}]), \quad \delta_i \sim \mathcal{N}(0,\ \sigma_i^2).$$

La desviación típica σ_i se ajusta automáticamente:

$$\sigma_i = \kappa(L)\,\Delta_i, \quad \Delta_i = x_i^{\text{máx}} - x_i^{\text{mín}}.$$

Donde,:

$$\kappa(L) = \kappa_{\text{mín}} \cdot + \kappa_{\text{máx}}\, \text{mín}\left(1, \tfrac{L}{S}\right),$$
$$\kappa_{\text{mín}} = 0{,}01, \quad \kappa_{\text{máx}} = 0{,}03, \quad S = 40.$$

Así, especies de vida más larga ($L \to S$) permiten mutaciones mayores, mientras que las de vida corta hacen ajustes más finos. κ son límites para σ y con ello la amplitud de las mutaciones. El S=40, es simplemente en función de mis datos, ya que es el tamaño medio de vida de un roble, especie con mayor lifespan en las pruebas.

DEAP dispone de una función de mutación Gaussiana - **pero no entiende de los rangos naturales de mis genes**, por eso es por lo que implementé un operador de mutación customizado, ya que mis fenotipos son heterogéneos (cada gen rangos diferentes). En dicho operador, utilizo la función Gaussiana del paquete *random*, biblioteca estándar de Python. He visto que hay formas de pasarle vectores a la función de mutación en DEAP, pero al final he preferido disponer de mi implementación.

Sea cual sea la función de mutación que se utilice, si se determina que un individuo ha de mutar, cada uno de sus genes tendrá un 0.15 de probabilidad de hacerlo; de ésta manera balanceo la distribución de genes mutados, pero sin fomentar que converja demasiado rápido.

3.2.5.7. Fase de acción

Fase final: Implementación de genotipos

1. Nuevas entidades

Una vez obtengo los genotipos evolucionados, genero las nuevas entidades e introduzco en el bioma.

2. Registro evolutivo

Todos los datos relevantes de la nueva generación (genes, métricas etc), los registro para análisis posterior y seguimiento de tendencias de adaptación.

3. Posible ajuste del ciclo

Por último, es posible que se ajuste el tiempo hasta la próxima generación, depende un poco de los resultados y condiciones ambientales.

Como nota extra, el número de descendientes creados es exactamente el valor `k_best` calculado anteriormente.

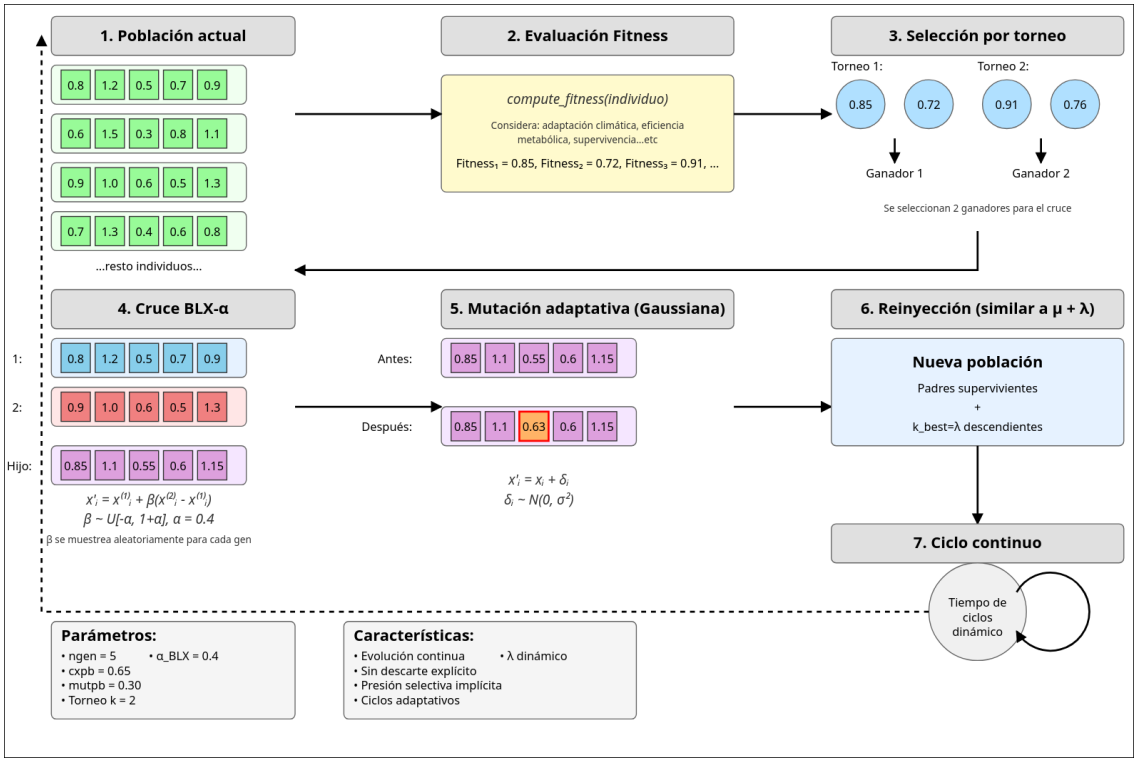


Figura 3.6: Modelo evolutivo, más en detalle. **Nota**: ngen=5, en cada ciclo. Datos numéricos son meros ejemplos

3.2.6. Control poblacional inteligente

Conforme los biomas iban cobrando vida, me resultaba cada vez más complicado mantener el balance y evitar extinciones. Así que pensé en implementar un sistema de control poblacional, pero basado en herramientas que he aprendido en la universidad y que antes nunca me habría atrevido a usar. Ahora es algo sencillo, pero cuando no se tienen las bases, no lo parece: **una regresión lineal con reglas por umbrales**.

Con ello en mente, quería ir un poco más allá. Había escuchado el término *fuzzy* o **difuso** en varios contextos (búsquedas de ficheros en terminal (fuzzy-finder llevo utilizando tiempo), en algoritmos...etc) y, tras investigar, vi que la lógica difusa era perfecta para suavizar los cambios abruptos al cruzar umbrales (ej: si la población pasa de 150 a 151 y tengo que decidir cuantas entidades instanciar, sin suavizado quizá me salte de 30 a 50 nuevas igualas, por ejemplo). Entonces,

esta con esta técnica puedo hacer transiciones suaves; me cuantifica el grado en que pasamos de *poco* a *mucho*. Me ha encantado aprender estos conceptos, me han dado muchas ideas. Bien, como hemos visto, durante el proceso evolutivo registro datos de genes y métricas. Con un histórico de las últimas 100 generaciones, analizo tendencias usando `np.polyfit`; he visto se usa mucho polyfit con grado 1 directamente para regresión lineal, y así lo he hecho. Ahora, para que la tasa de cambio sea comparable independientemente del tamaño de población, normalizo la pendiente: $\tilde{m} = \frac{m}{P_n}$.

Con ello, convierto la pendiente en una **tasa de crecimiento relativa**, y así puedo evitar problemas con poblaciones de diferentes tamaños, como al principio me ocurría.

Lógica difusa con `scikit-fuzzy`

Gracias a este tipo de lógica, puedo definir reglas que establecen asociaciones del tipo *si la pendiente es . . . entonces el ajuste es . . .*. Por ejemplo:

Rule 1: `severe_decline` ⇒ `large_increase`,
Rule 5: `rapid_growth` ⇒ `reduction`.

1. Universo de discurso

Para mi sistema sólo necesito dos magnitudes:

`slope` $(\dot{N}/N) \in [-1, 1]$ y `adjustment` $A \in [0{,}5, 2{,}5]$.

El primero es la *pendiente normalizada* y me indica la velocidad de cambio poblacional en un instante. El segundo es el *factor de ajuste*, que realmente utilizo a modo de **amplificador ya que multiplicará la población** en la siguiente generación. Entre ambos intervalos, se constituye el denominado *universo de discurso* - el rango numérico donde definimos los conjuntos difusos, básicamente.

Tanto para realizar los cálculos como para definir las cinco etiquetas difusas, he utilizado el motor que ofrece la librería de **scikit-fuzzy**. Utilizo las funciones de `trapmf` (trapecio) y `trimf`(triángulo). Rangos:

- `severe_decline`: −0,15 – −0,07 (trapecio)
- `moderate_decline`: −0,08 – −0,01 (triángulo)
- `stable`: −0,02 – 0,04 (triángulo)
- `moderate_growth`: 0,03 – 0,12 (triángulo)
- `rapid_growth`: 0,10 – 0,30 (trapecio)

2. Conjuntos de pertenencia

En lugar de decir «la pendiente vale −0,06» ahora podemos decir «tengo un *moderate_decline* con grado 0.8». Esa graduación de *cuánto* viene dada por **cinco funciones de membresía** (fig. 3.7).

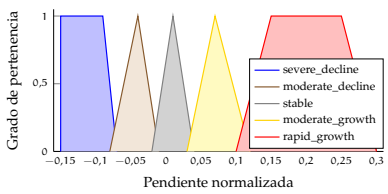


Figura 3.7: Funciones de membresía difusa para la pendiente poblacional.

Los trapecios en los extremos, son para mantener membresía o pertenencia = 1 en un `plateau` central; esto evitar que un error numérico pequeño cambie drásticamente la etiqueta (como ocurriría si únicamente utilizamos reglas o *ifs* con umbrales fijos).

3. Reglas: de la pendiente al ajuste o amplificación.

Las reglas capturan la intuición de `si sube mucho, frena; si baja, empuja`:

R1: `severe_decline` ⇒ `large_increase` (1.8–2.5)

R2: `moderate_decline` ⇒ `moderate_increase` (1.4–1.9)

R3: `stable` ⇒ `small_increase` (1.1–1.4)

R4: `moderate_growth` ⇒ `small_increase` (1.1–1.4)

R5: `rapid_growth` ⇒ `reduction` (0.5–0.95)

Las reglas R_3 y R_4 hacen también ajustes pequeños hasta cuando es bastante estable; sirven para amortiguar volatilidad con ello. R_5 es básicamente un freno.

4. ¿Para qué sirve A?
Como he mencionado anteriormente, el valor devuelto A es un *amplificador* con el que escalo el parámetro `k_best`. Es decir, según la tendencia de su propia especie, el agente evolutivo amplía (o reduce) el espacio de búsqueda y decide cuántos genotipos *buenos* se instancian en el bioma.
No obstante, como más adelante se podrá apreciar en el tema del Agente Equilibrium (sistema neurosimbólico), ésta funcionalidad de *smart population control* ha quedado un poco obsoleta. Pero he aprendido mucho, me ha parecido interesante y por ello he considerado exponerla también.

3.2.7. Análisis de tendencias evolutivas

En el sistema evolutivo también he incluído un **módulo de análisis de cambios genéticos a lo largo del tiempo para detectar tendencias de adaptapción**. La idea es registrar cómo cambian los rasgos genéticos de flora y fauna conforme pasan las generaciones.

3.2.7.1. Registro generacional

Para cada generación y especie, registro lo siguiente:

- Valores medios de rasgos genéticos
- Tamaño de la población
- Distribución de valores genéticos

3.2.7.2. Input de datos

Al analizador le paso los siguienes historiales como input:

- **Historial de flora**: estadísticas de genes; eficiencia fotosintética, resistencias térmicas, tasas de absorción de nutrientes… etc.
- **Historial de fauna**: Es más escueta, principalmente características de adaptación térmica.

3.2.7.3. Proceso de análisis

Una vez más, utilizo regresión lineal para detectar tendencias. Sigo el siguiente proceso:

1. Para cada rasgo, por especie y generación calculo las medias.
2. Trazo en una línea temporal.
3. Necesito la pendiente normalizada, la calculo:

$$\text{Pendiente} = \frac{\Delta(\text{gen})}{\Delta(\text{generación})}.$$

4. Con ella calculada, la asocio al gen/rasgo.
5. La magnitud de cambio es el valor absoluto de la pendiente.
6. Dirección del cambio, simplemente crec. si pendiente > 0; decr. si < 0.
7. Con ello monto tuplas `(tipo, especie, gen, magn., dir.)`.
8. Las ordeno por magnitud de cambio descendiente
9. Ahora selecciono las 3 más fuertes para mostrar.

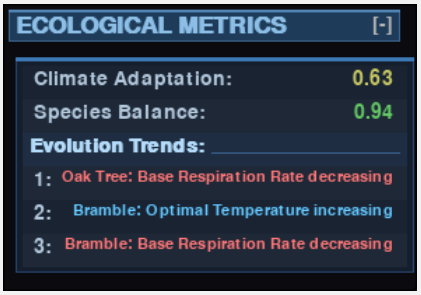


Figura 3.8

3.2.8. Integración con otros sistemas

El agente evolutivo interactúa con muchos componentes del bioma y crea bucles de retroalimentación - un intento de hacer que el modelo de simulación sea un eco de dinámica de sistemas:

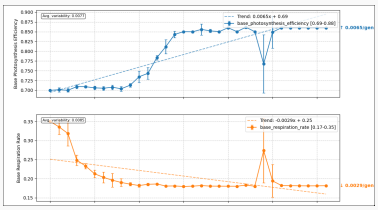
- **Sistema climático**: Como hemos visto, recibe datos climáticos que influyen en las funciones de fitness. Pero, a su vez, la evolución de la flora afecta indirectamente al clima; cambios en la absorción de CO_2, transpiración etc.

- **Agente de Equilibrium**: Le proporciona datos sobre tendencias evolutivas que pueden ayudar a tomar decisiones de intervención ecológica, como se verá más adelante.
- **Sistema de coevolución depredador-presa y flora**: Depredadores, presas y plantas se adaptan en cadena: mejoras en la caza mejoran defensas en las presas, lo que a su vez lleva a las plantas a reforzarse (toxinas, cambios en floración/crecimiento, etc). Se podría decir no muy alto que cuando uno avanza crea presión selectiva en los demás.

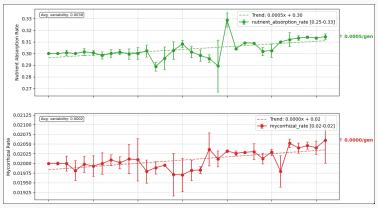
Ecos en el tiempo

Vamos a ver cómo se refleja el paso de la evolución en el bioma.

Evolución de genes / traits



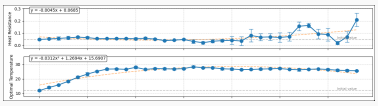
En las gráficas se puede ver la evolución de los diferentes rasgos genéticos - los patrones tienen bastante sentido con las presiones selectivas del entorno. La **eficiencia fotosintética** (azul, derecha) tiene una tendencia ascendente clara desde ≈0.70 hasta estabilizarse alrededor de 0.85; quizá converge un poco más rápido de lo que esperaba, pero poco a poco va efectivamente subiendo como debe.



La **tasa de respiración base** (naranja) hace justo lo contrario - cae desde 0.35 hasta ≈0.18, lo cual está bien ya que existe una correlación negativa con la eficiencia fotosintética. Se ve claramente el efecto de la función de fitness y cómo ésta correlación se moldea.

La **tasa de absorción de nutrientes** (verde) se mantiene bastante estable con ligeras fluctuaciones, y que la tasa micorrícica (rojo) va por su cuenta con variabilidad *natural*.

Caso especial - Adaptación térmica:



Para los datos de temperatura (azul, derecha) tengo un caso particular - un `mushroom` con temperatura óptima inicial de 10°C que intoduje de manera forzada / artificial en un bioma tropical. Poco a poco se ve cómo se va adaptando; su temperatura óptima sube gradualmente desde ≈15°C hasta cerca de 25°C, y además adquiere resistencia al calor. No es el mejor ejemplo de adaptación térmica, pero ilustra bien cómo efectivamente, el sistema evolutivo responde a presiones ambientales extremas con ajustes graduales en los genes relevantes.

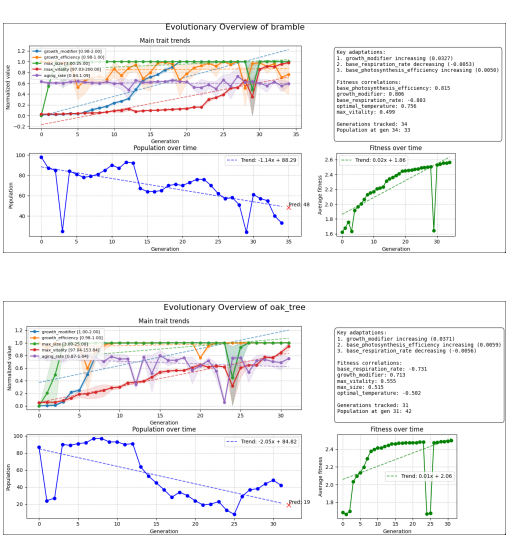


Figura 3.9: Paisajes evolutivos, primera versión.

Un pequeño despiste en el tracker

Antes de comentar, me gustaría especificar que he dejado estas gráficas porque tienen una relación me ha parecido interesante, sin embargo en ese momento tenía un pequeño **bug** en el sistema de tracking - que ya he corregido - la función de fitness muestra al último individuo en ser evolucionado para cada generación (una vez corregido, es la media de esa generación en su lugar).

El problema que ocurre es que si hay varianza grande en los antecesores, ya existe más probabilidad de que el último individuo en evolucionar sea uno de los que poseen *malos genes*; la selección por torneo puede escoger a padres con bajo fitness. No obstante, salvo ese detalle, las gráficas de fitness muestran patrones evolutivos coherentes con las presiones selectivas.

En general, todas las especies muestran una **tendencia ascendente** - el `deer` alcanza valores de fitness alrededor de 9-10, el `mushroom` y `oak_tree` se estabilizan cerca de 2.4-2.5. Lo que me ha parecido un punto notable es el poder observar cómo, cuando la varianza / diversidad incrementa, cae en picado el fitness, pero además ha coincidido con caídas de población de la especie.

Versión corregida - Mejoras observadas:

Las gráficas que se muestran en la figura (3.14) han sido generadas con la versión corregida. Si nos fijamos en general - podemos ver un comportamiento más **suave y estable**, con un crecimiento gradual. Las correlaciones de fitness que se ven en las tablas tienen sentido - por ejemplo, en `oak_tree` la correlación negativa de `base_respiration_rate` (-0.731) con fitness nos dice que respirar menos / más lento es mejor para la supervivencia, y que `growth_modifier` (0.713) y `max_vitality` (0.555) tienen correlaciones positivas - como era de esperar.

Los `brambles` muestran un patrón similar al `mushroom` - su fitness sube de manera consistente hasta 2.6. En general, se ve que el sistema evolutivo funciona; las especies se adaptan y mejoran su fitness a lo largo del tiempo - salvo por las caídas explicadas.

Los `trait trends` que se ven arriba a la izquierda de cada gráfica son las **medias por generación**, y las bandas/áreas translúcidas son la desviación estándar. Esta varianza, se nota mucho en el `mushroom` y `deer` - tienen bandas de incertidumbre muy amplias que implica demasiada diversidad genética a partir de la generación 18 y 60 respectivamente; que casualmente coincide con los picos o caídas de fitness.

Correlaciones Oak Tree	`base_respiration_rate`	`growth_modifier`	`max_vitality`
Valor	-0.731	+0.713	+0.555

Interpretación: Tasa de respiración baja mejora supervivencia (se asocia a menor estrés, una mejor eficiencia fotosintética etc.); crecimiento y vitalidad correlacionan positivamente.

Post-Corrección

Nota: Tras haber corregido el error mencionado, la diferencia es notable. No obstante, muy de vez en cuando se puede observar alguna caída súbita en fitness (he realizado muchas pruebas y en pocas ocurre) - mucho más suave y más puntual ahora que es la media de la nueva generación. Creo que ésta patología es fruto de la combinación de **3 factores**: baja población, selección por torneo y el solapamiento de generaciones que tienen los biomas. Puede existir una mala generación que, por azar o deriva genética, propague temporalmente genotipos *malos* hasta que se recombinen y se termine restaurando el óptimo.

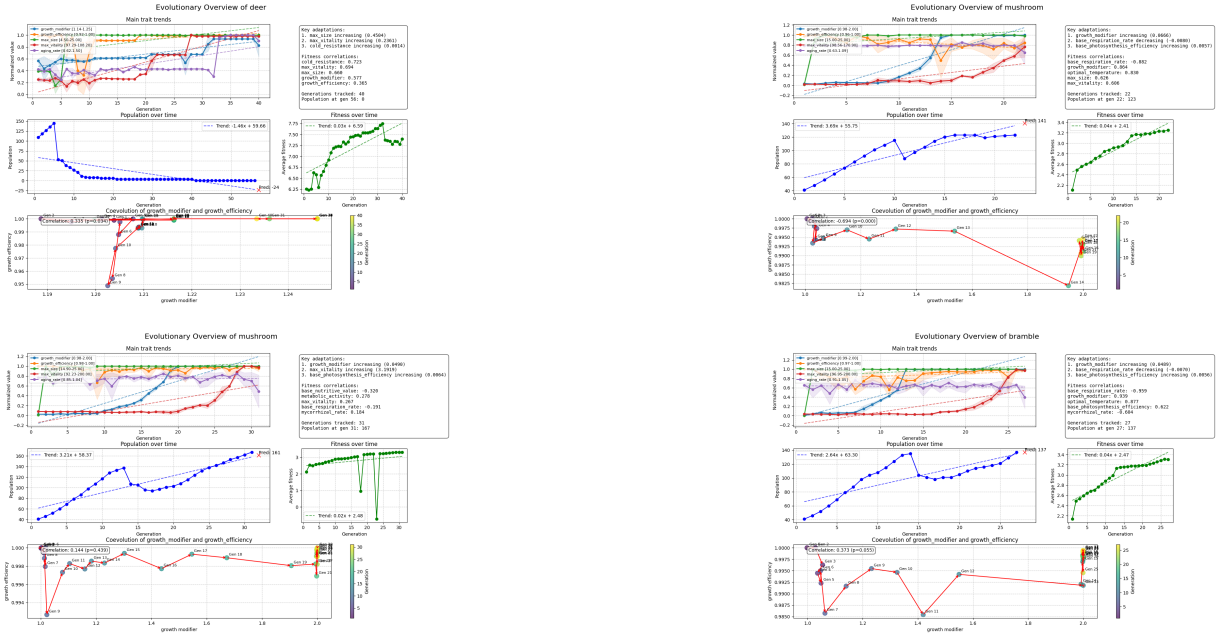


Figura 3.14: Paisajes evolutivos corregidos

Veamos la misma especie (**desplazada**) tratada al final del tema anterior - **oaktree**, con su lifespan de 40 años y temperatura óptima (inicial) de 28ºC. Vamos ver a la evolución de la especie a lo largo de 10 eras con 100.000 eventos cada una (**1.388 años - 105 generaciones de oaktrees**) y en **Tundra**:

Figura 3.17: Fitness (izquierda) y evolución de algunos genes (derecha)

Figura 3.20: (Ampliar documento) Simulación de 1.388 años cargada en el visor. Izquierda generación 0 y derecha generación 103 de **oaktree**. Es normal ver el estado *unstable*, ya que ésta prueba ha sido específica para observar ésta especie. **Recordemos que esta especie no corresponde a la Tundra**.

Es apasionante ver como, efectivamente, se ha ido adaptando al clima de la tundra, y generaciones avanzadas han ido adquiriendo una temperatura óptima base inferior, además de resistencia al frio. El que los genes evolucionen y también mejoren la resistencia al calor, es normal ya que esto se produce cuando la temperatura base es menor pero la temperatura en estaciones como verano puede llegar a los 5-8º. **Nota**: observemos algo precioso en imagen derecha; al adaptarse - el aging rate es inferior a 1.0 incluso a estas alturas de su vida. Además, si nos fijamos ha incrementado su lifespan y el específico seleccionado en imagen derecha, consigue vivir hasta los ≈58 años.

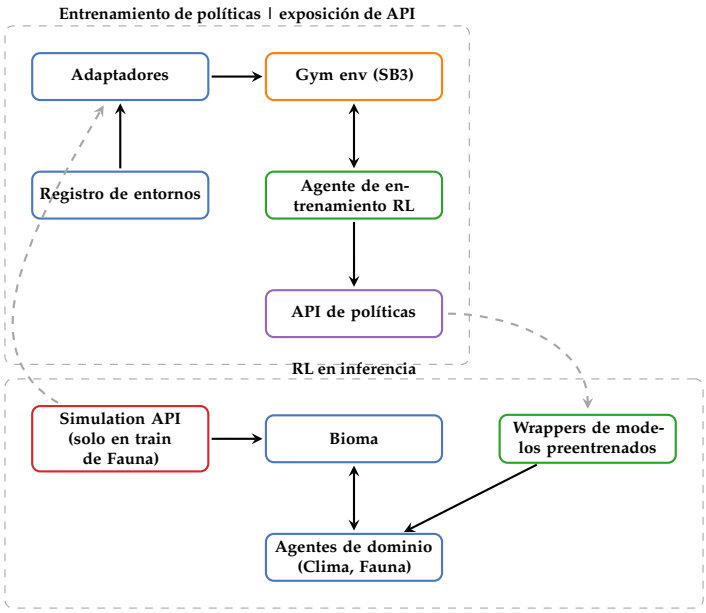
3.3. Agentes de Reinforcement Learning

En el módulo de `research` (mencionado en el documento anexo de fundamentos y arquitectura), he preparado un entorno de entrenamiento de Reinforcement Learning. Para la gestión de algoritmos, utilizo el framework de Stable Baselines 3 (SB3). Lo conocí en la asignatura ATAI *Advanced Techniques of Artificial Intelligence* de 4º y ofrece soporte completo para Gym, creado por OpenAI y ahora fork como Gymnasium por Farama Foundation.

3.3.1. Arquitectura: adaptadores, entornos e inferencia

En `research`, tanto para Deep Learning como para Reinforcement Learning, existen sistemas de entrenamiento/ búsqueda de políticas óptimas e interfaces de integración para inferencia. Creo que uno de los sistemas más importantes son los adaptadores; he utilizado el patrón Adapter para aislar cada entorno según su modelo - es una especie de sistema-puente entre el simulador y los frameworks de aprendizaje. Cada uno de los agentes dispone de su propio adaptador.

Ahora, una vez encontradas las políticas óptimas, han de ser accesibles en **modo inferencia**, así que he diseñado la arquitectura para que ofrezca soporte al Bioma y sus subsistemas para hacer uso de los modelos preentrenados durante la simulación. Veamos un pequeño diagrama:



Simulación normal/Bioma	Adaptadores / Entorno RL
Tiempo realista/automático	Tiempo guiado por RL environent, (solo en fauna)
Configuración estática	Configuraciones aleatorias
Bioma completo interconectado	Fragmentos de bioma aislados
Agentes/Wrappers de modelos	Agentes de entrenamiento

En la tabla muestro una simulación normal (sin entrenamiento) vs un entrenamiento y cómo los adaptadores gestionan las cosas. **El objetivo final de todo esto** es dar soporte para que cada agente de dominio (clima, fauna, equilibrium), ejecute sus decisiones de forma totalmente autónoma pero, basándose en unas políticas aprendidas; los modelos preentrenados.

Entrenamiento de políticas:

- Con los adaptadores he creado una especie de sandbox virtual - transformo los sistemas para que funcionen con entornos Gymnasium. He escogido Gymnasium, porque me ofrecen una interfaz estándar que puedo usar con todos los algoritmos de RL.
- El núcleo de ésta parte, es el agente de entrenamiento buscando políticas óptimas e interactuando con dicho entorno.
- La API de políticas es una capa que expone funcionalidades y esconde los intríngulis del modelo; así queda accesible durante la fase de inferencia.

RL en inferencia:

- La Simulation API controla la ejecución global y proporciona los datos para el entrenamiento (esto ocurre solo durante el entrenamiento de Fauna).
- **Como punto central** tengo el **bioma** - es donde coexisten los agentes.
- Para utilizar las políticas ya aprendidas, hay unos wrappers que integran funcionalidad de inferencia + lógicas específicas a la simulación (todos los `..*AgentAI`)
- Con todo esto, cada agentes de dominio ya puede ejecutar sus decisiones de forma autónoma según las políticas que han aprendido.

3.3.2. Agente climático

Primero, **vamos a ampliar el contexto del sistema climático que hemos visto** - hasta ahora, sabemos que es un sistema que dispone de diversos factores ambientales que se modifican durante la simulación y, que estos afectan al estado general del bioma (y a las entidades). También hemos visto que se mantiene un historial con todos los cambios, que otros agentes y sistemas de métricas utilizarán. Recordemos brevemente **su integración con el agente evolutivo**; a medida que las condiciones climáticas cambian, se va ejerciendo presión selectivas sobre las especies, y esto favorecía adaptaciones que mejoren la tolerancia a las nuevas condiciones.

Como se explicó en el tema anterior, el sistema climático está diseñado tal que disponga de bucles de retroalimentación complejos; la densidad de vegetación (biomasa) influye en los niveles de CO_2, que a su vez afectan poco a poco a la temperatura global. Otro ejemplo puede ser la actividad metabólica de los organismos; ésta contribuía a cambios en humedad a través de procesos como la transpiración en flora.

Entonces, aunque hayamos visto cómo se modifican y qué les afecta, no hemos visto **de dónde vienen esos valores de los factores medioambientales**, ni tampoco qué los producen ni cómo repercuten los eventos meteorológicos. Así que, veamos sus pilares:

Componentes principales del sistema climático

- **ClimateSystem**: *Motor* climático; tiene varias tareas, pero principalmente calcula los factores dependiendo del estado del bioma y las entidades, gestiona susbsistemas como el estacional, hace handling de diferentes eventos etc.
- **ClimateState** - Estructura de datos; valores de todos los factores ambientales (temperatura, humedad, niveles de CO_2, densidad de biomasa, presión atmosférica, etc), incluso el estado meteorológico actual.
- **SeasonSystem** - Control sobre el calendario del bioma y estación actual.
- **WeatherEvent** - Fenómenos como tormentas, sequías, lluvías, días soleados...etc.

He definido un conjunto de rangos de comfort base para los factores ambientales de cada tipo de bioma/estaciones (`ecosystem.json`. Son valores orientativos, pero están basados en diferentes fuentes (ejemplo)):

Bioma	Estación	Factor	Mínimo	Máximo
Tropical	Primavera	Temperatura (°C)	24	29
Desértico	Verano	Humedad (%)	2	15
Taiga	Invierno	Temperatura (°C)	-45	-10
Sabana	Otoño	Precipitación (mm)	20	60
Tundra	Verano	Temperatura (°C)	2	15

Tabla 3.1: Ejemplo simplificado de rangos de confort en distintos biomas y estaciones

3.3.2.1. Justificación del enfoque de RL

En este punto, me gustaría justificar el por qué he escogido Reinforcement Learning para un sistema climático, ya que entiendo puede resultar un poco extraña la elección. Sé que están muy extendidas técnicas de forecasting con deep learning para predecir estados futuros, y en muchos casos se agregan capas extra que en base a esas predicciones se tomen las acciones oportunas - esto requiere capa y complejidad extra, sin embargo en RL **se aprende directamente la política** $\pi(a \mid s)$ **que mapea estado** $\rightarrow$ **acción**.

Lo primero en lo que pensé al enfrentarme a este problema, fué en **los datos**. Yo no dispongo de grandes cantidades para entrenar un modelo mediante Deep Learning e investigando, no conseguí encontrar datasets climáticos que contengan datos de factores medioambientales organizados por bioma. Podría intentar crearlos yo, implementar un pipeline de descarga + preprocesamiento para mezclar diversas fuentes y generar los datasets con el formato que yo quisiera, pero incluso para ese caso es probable que me hicieran falta herramientas y técnicas extra como rasters de variables climáticas para extraer información... determiné que el tiempo que me llevaría, no merecía la pena.

Si nos fijamos en la tabla 3.1, se puede ver que podemos intentar que el agente aprenda políticas de manera que, dado un bioma y una estación, **se centre en que los factores medioambientales estén en sus rangos de comfort**. El RL tiene como fundamento el interactuar con el entorno, así que el inconveniente podría ser la gran cantidad de interacciones que necesita - pero, al disponer de un framework altamente modular, **puedo fragmentar el bioma y extraer únicamente el sistema climático de manera totalmente independiente**, y utilizarlo para que el agente interactúe con el estado climático - esto lo hace de un coste computacional muy bajo.

Además, existe una idea que me gustó mucho cuando estudié Reinforcement en la asignatura de ATAI - podemos hacer que el modelo aprenda a planificar y trazar estrategias para conseguir el máximo reward acumulado - esto implica que no siempre opte por la mejor opción (*greedy*), esto lo hacemos estableciendo el factor de descuento (γ) a un valor alto, para que aprenda a valorar rewards futuros. **Esto es muy útil en los cambios de estaciones**, como se verá un poco más adelante.

Por último, gracias a RL podemos optimizar **objetivos múltiples** (varios factores medioambientales) y no lineales cuando exploramos el entorno con rewards en lugar de requerir una función de pérdida compuesta y diseñada de forma muy manual. Es decir, podemos guiar de manera sencilla y natural al modelo mediante rewards para que aprenda patrones al desviarse en temperatura ($\Delta T = 1\,^{\circ}C$) o en humedad ($\Delta H = 5\,\%$), por ejemplo; la política aprende automáticamente el equilibrio óptimo entre ambos - saca conclusiones sobre el **trade-off** por si misma.

3.3.2.2. Observaciones heterogéneas, acciones y cambio de estaciones

El agente climático opera sobre los siguientes espacio de observación y acción:

Observation Space

Temperature	$\in [0, 1]$[1], `float32`
Humidity	$\in [0, 1]$[1], `float32`
Precipitation	$\in [0, 1]$[1], `float32`
Biome Type	Discrete (número de tipos de biomas)
Season	Discrete (número de estaciones)

(Normalización Min–Max)

WeatherEvent actions

- `clear`
- `sunny`
- `partly_cloudy`
- `cloudy`
- `fog`
- `drizzle`
- `light_sprinkle`
- `rain_shower`
- `rain`
- `heavy_rain`
- `thunderstorm`
- `snow`
- `blizzard`
- `heatwave`
- `drought`
- `windy`

Ésta versión del entorno, ha sido llamada `NaiveClimateEnvironment` ya que es una versión bastante simplista, aunque el usuario puede introducir más parámetros de manera sencilla y reentrenar. No obstante, ahora mismo cada evento meteorológico modifica las variables climáticas (temperatura, humedad, precipitación), con rangos de deltas predefinidos pero con **variabilidad estocástica**:

Example Weather Effects

`partly_cloudy`	*Temperature*: [1,0, 1,5], *Humidity*: [0,0, 0,5], *Precipitation*: [0, 0]
`sunny`	*Temperature*: [4,0, 4,5], *Humidity*: [−6,0, −5,5], *Precipitation*: [−1,0, −0,5]
`light_sprinkle`	*Temperature*: [−0,5, 0,0], *Humidity*: [2,0, 2,5], *Precipitation*: [3,5, 4,0]
`fog`	*Temperature*: [−2,0, −1,5], *Humidity*: [11,5, 12,0], *Precipitation*: [0,5, 1,0]
`heavy_rain`	*Temperature*: [−4,5, −4,0], *Humidity*: [16,0, 16,5], *Precipitation*: [35,0, 35,5]
`clear`	*Temperature*: [2,0, 2,5], *Humidity*: [−3,5, −3,0], *Precipitation*: [0, 0]

Esto significa que si el modelo nos devuelve la acción sunny, aplicará los deltas asociados a sunny al estado del clima actual. Es una vertiente sencilla y no muy realista, pero para prototipar un clima que varíe, planifique y sea adaptativo, ha funcionado bastante bien. Veamos un ejemplo en la UI:

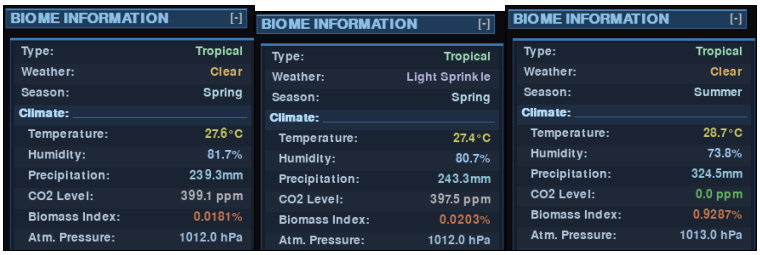


Figura 3.21: Ejemplos de weather y factores medioambientales. **En la realidad, los cambios en el clima producen los eventos meteorológicos**, pero este enfoque inverso consigue un efecto parecido en la simulación.

Ahora que hemos visto cómo influye la ejecución de un evento meteorológico, explico brevemente qué implica un cambio de estación. La idea es muy parecida a lo que acabamos de ver. **El sistema climático, irá dirigiendo al sistema de estaciones para que vaya avanzando** y, cuando corresponda, cambie de estación (cada ≈ 90). En el momento en el que entre la nueva estación, se aplican unos deltas con respecto al estado actual de los factores medioambientales. Veamos unos pocos ejemplos:

Bioma	Estación	ΔTemp. (°C)	ΔHum. (%)	ΔPrec. (mm)	ΔPres. (hPa)
tropical	spring	0	5	500	-2
desert	summer	7	-10	-20	5
taiga	winter	-13	-15	-150	7
savanna	autumn	-1	-10	-300	1

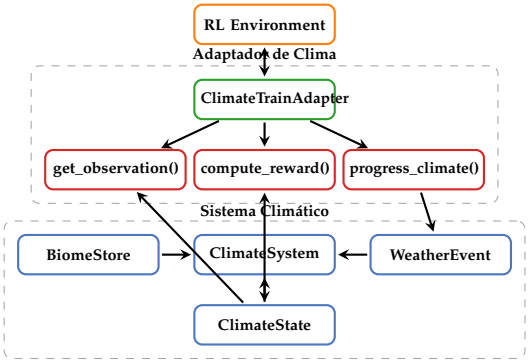
Puede que al ejecutarse un cambio de estación se realicen cambios bruscos - como el de la caída de temperatura en la Taiga al entrar el invierno. Es brusco, pero con entrenamiento, el **modelo aprenderá a predecirlo y planificará de antemano estos cambios, con lo que se irá adaptando gradualmente**.

3.3.2.3. Diseño del adaptador de entrenamiento

Vamos con el primer adaptador especializado: `ClimateTrainAdapter`. `ClimateTrainAdapter` actúa como una especie de interfaz entre el agente de RL y la simulación; primero traduce las acciones discretas que nos devuelve el modelo, en eventos meteorológicos y luego lo aplica al estado general pero - **lo bonito de estos sistemas es que encapsulan todo el estado climático que se necesita para aprender sin interferir con el bioma real (ni necesitar de él, si quiera)**.

Responsabilidades

1. La primera es el inicializar un *micro-clima* de un bioma en concreto que nos diga el entorno de `gym`.
2. Cuando el agente determine una acción (índice de *WeatherEvent*), convertirla en una llamada al **ClimateSystem** para que ejecute esa acción y ésta modifique el entorno.
3. Obtener el nuevo estado como observación **compatible** con `gym`.
4. También calcula el reward, se basa en la proximidad a rangos de confort que se hayan definido para el bioma y estación actuales. Más sobre esto en su apartado correspondiente.



Como se puede observar, tenemos una **gran ventaja gracias a este aislamiento**; sólo se carga el estado climático con factores mediambientales; flora, fauna y el resto de módulos del bioma están inactivos - no hay nada más del bioma, tenemos una especie de *sandbox* de usar y tirar, como se verá en el proceso de entrenamiento.

Figura 3.22: Flujo del Adaptador de Clima para entrenamiento con RL

3.3.2.4. Optimización de políticas: selección del algoritmo

Cuando tuve que decidir qué algoritmo utilizar para buscar una política óptima, tenía claro iba a utilizar Q-Learning pero con DQN (Deep Q Network) para el agente de fauna, pero aún así quise explorar diferentes opciones.

En la asignatura de ATAI conocí brevemente PPO (Proximal Policy Optimization) y recordaba que funciona bien en entornos con cambios bruscos. Investigué más sobre este algoritmo y, efectivamente, PPO implementa un mecanismo de clipping que evita cambios drásticos en la política durante la optimización. Me pareció ideal, ya que, en mi entorno, el clima sufre variaciones por eventos meteorológicos y cambios de estación - lo que puede llegar a generar transiciones bruscas.

(*Schulman et al., 2017*[44]) destaca precisamente esta idea; la documentación de Stable Baselines 3 también lo menciona (*Stable Baselines3 Contributors, 2025*[45]): "The main idea is that after an update, the new policy should be not too far from the old policy. For that, ppo uses clipping to avoid too large update.".

Otra particularidad que me gustó de PPO es que posee capacidad para manejar espacios de acción grandes - mi implementación actual solo tiene 16 acciones, pero si en el futuro necesito expandirlas, PPO se adaptaría sin problemas porque simplemente requiere ensanchar la capa softmax final. En contraste, DQN tiene problemas con espacios de acción amplios ya que necesita una cabeza Q(s,a) adicional por cada acción; y esto, implica tener que recalcular Q(s,a) por cada neurona en capa de salida (*Mnih, Kavukcuoglu, Silver, Rusu et al., 2015*[46]; *Schulman et al., 2017*[44]).

1. Análisis del problema

- **Estado:** continuo (temperatura, humedad, precipitación **normalizados**) + índices de bioma y estación.
- **Acciones:** discretas, pocas (índices de `WeatherEvent`).
- **Rewards:** densa (doy señal en cada paso), suave y no lineal (sigmoide); mezclo rangos de confort y penal. por extremos.
- **Dinámica:** no estacionaria (por cambio estacional y diferentes biomas); pero marco con variables/índices; con esto evito cambios bruscos.
- **Coste de simulación:** bajo ⇒ puedo generar tantos episodios como quiera (online RL viable).

El framework de SB3, independientemente del algoritmo que se utilice dispone de soporte para políticas `MultiInputPolicy` - con ello, instancia distintos extractores dependiendo del tipo de datos que le pase como observación, y los combina al final. Más detalles, en el agente de Fauna.

2. Puntos-clave para escoger algoritmo

No obstante, he reordenado las comparativas entre PPO/DQN teniendo como pivote los puntos clave en los algoritmos de RL, llevados a mi terreno y particularizados para mis simulaciones, en esta tabla: **Comparativa de puntos clave de PPO/DQN en la simullación** (Apéndice, p. 124).
Esto es la teoría que me hizo llegar a la conclusicón de que **PPO era el algoritmo que más me convenía**, hasta que los probé de manera conjunta.

Pruebas empíricas
Lo importante es comparar PPO vs DQN, pero ya que SB3 dispone de un API muy flexible y sencilla, he incluido A2C para contrastar resultados. Expongo hiperparámetros bastante estándar para los 3 a modo de realizar una prueba genérica en mi entorno:

Comparativa de algoritmos

PPO	DQN	A2C
learning_rate: 3e-4	learning_rate: 3e-4	learning_rate: 3e-4
n_steps: 2048	buffer_size: 100 000	n_steps: 5
batch_size: 64	batch_size: 64	gamma: 0.99
n_epochs: 10	gamma: 0.99	gae_lambda: 1.0
gamma: 0.99	exploration_fraction: 0.1	ent_coef: 0.02
ent_coef: 0.02	exploration_initial_ep 1.0	
	exploration_final_eps: 0.05	
	target_update_interval 1000	

Algoritmo	Recomp. Media	Recomp. Std	Recomp. Máx	Recomp. Mín	Long. Media	Total Epis.	Converg.
PPO	5075.45	2618.40	9058.27	−4195,41	720.0	1390	0.02
DQN	6064.68	2906.74	9282.63	−4587,27	720.0	1388	0.01
A2C	2739.50	2725.65	8493.63	−4258,96	720.0	1388	0.21

Visto que parecía se podrían beneficiar de más entrenamiento, realicé uno de 1.000.000 de timesteps, veamos los resultados:

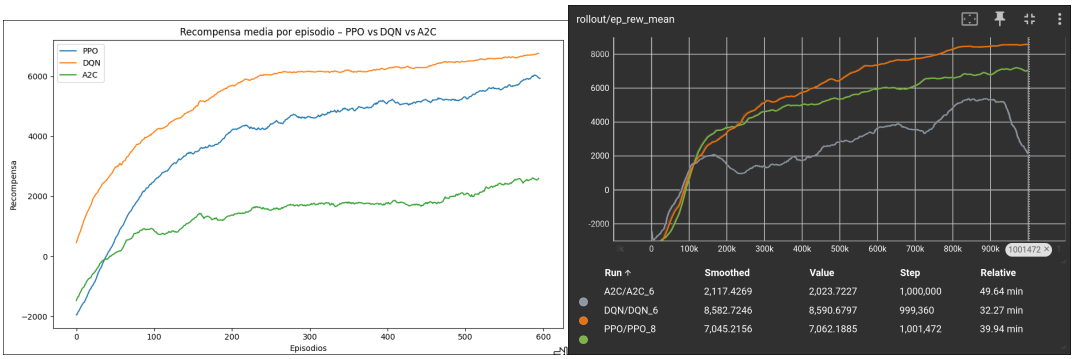


Figura 3.23: Hice el entrenamiento sin llamar a la función de plotting con matplotlib por error, pongo Tensorboard.
(Derecha) 1 millón de steps. **Nota**: Cambia el color verde de algoritmo entre gráficas.

Los hiperparámetros que mejor me han funcionado son bastante estándar. Con el que he jugado un poco más es con el coeficiente de entropía, para que el agente explore un poco más en el caso de PPO, y los exploration rates en DQN. Aquí simplemente unas grafícas complementarias:

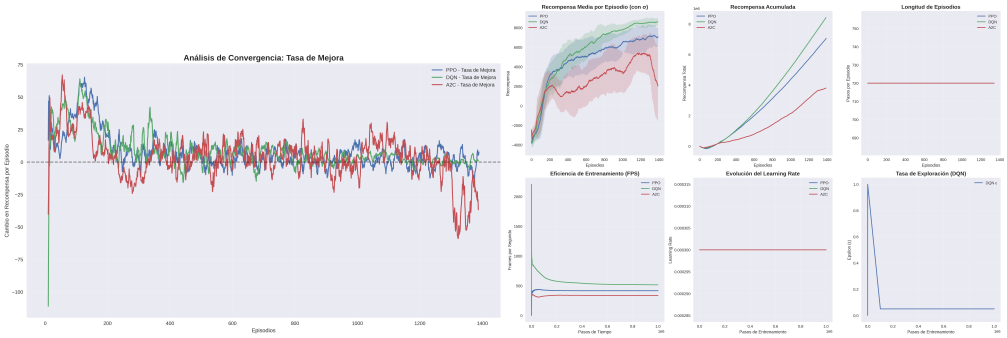


Figura 3.24

Nota: Para los entrenamientos de Reinforcement Learning, me baso principalmente en la métrica de **reward medio por episodio**, tanto aquí, como en el agente de fauna.
No podía estar más equivocado, DQN tiene un mejor rendimiento desde el principio, aunque la brecha se reduce con los episodios; ha funcionado mejor de lo que me esperaba.

Creo que uno de los posibles factores que me hizo determinarme por PPO incorrectamente, ha sido la idea de que los cambios de estación - y con ello, fluctuaciones en factores medioambientales - sean considerados cambios drásticos, quizá fué algo ingenuo por mi parte; ya que esos cambios igual no son tan violentos una vez **normalizados**, asi que quizá es trivial para cualquiera de estos algoritmos. Con ello, la ventaja del PPO debido al clipping se reduce.

Conclusión: Voy a mantener PPO como segunda referencia (es estable y se va acercando a DQN con los episodios), pero entreno DQN como modelo principal. Si el espacio de acciones crece o la dinámica se vuelve mucho más volátil, se puede hacer el cambio de manera muy sencilla a PPO.

3.3.2.5. Función de rewards

He diseñado la función de rewards $R(s, a)$ para enseñar al modelo que ha de mantener parámetros climáticos dentro de los rangos de comfort/óptimos. A grandes rasgos, la estructura de la recompensa es la suma de un componente base que se fija en si el parámetro está en su rango correcto, otro componente de penalizaciones basadas en distancia por no estarlo y el último, que otorga penalizaciones por valores extremos:

$$R(s, a) = R_{\text{base}} - P_{\text{distancia}} + P_{\text{extremos}} \tag{3.3}$$

Reward base:

$$R_{\text{base}} = \begin{cases} 6{,}0 & \text{si } T \in [T_{\text{mín}}, T_{\text{máx}}] \\ 0 & \text{en otro caso} \end{cases} + \begin{cases} 3{,}0 & \text{si } H \in [H_{\text{mín}}, H_{\text{máx}}] \\ 0 & \text{en otro caso} \end{cases} + \begin{cases} 5{,}0 & \text{si } P \in [P_{\text{mín}}, P_{\text{máx}}] \\ 0 & \text{en otro caso} \end{cases} \tag{3.4}$$

donde T, H y P representan la temperatura, humedad y precipitación actual respectivamente, y los rangos $[X_{\text{mín}}, X_{\text{máx}}]$ son los rangos de confort para una estación en un tipo de bioma.

Penalización por distancia:

La penalización por distancia será la suma de las **penalizaciones individuales por distancia de cada parámetro que no esté en su rango óptimo**:

$$P_{\text{distancia}} = \text{mín}\left(10{,}0,\ P_{\text{temp}} + P_{\text{hum}} + P_{\text{prec}}\right) \tag{3.5}$$

donde cada una es una sigmoidea:

$$P_{\text{temp}} = \begin{cases} 2{,}0 \cdot \left[\dfrac{1}{1+\exp(-0{,}25 \cdot d_{\text{temp}})} - 0{,}5\right] \cdot 2, & \text{si } T \notin [T_{\text{mín}}, T_{\text{máx}}] \\ 0, & \text{si } T \in [T_{\text{mín}}, T_{\text{máx}}] \end{cases} \tag{3.6}$$

$$P_{\text{hum}} = \begin{cases} 1{,}5 \cdot \left[\dfrac{1}{1+\exp(-0{,}15 \cdot d_{\text{hum}})} - 0{,}5\right] \cdot 2, & \text{si } H \notin [H_{\text{mín}}, H_{\text{máx}}] \\ 0, & \text{si } H \in [H_{\text{mín}}, H_{\text{máx}}] \end{cases} \tag{3.7}$$

$$P_{\text{prec}} = \begin{cases} 1{,}5 \cdot \left[\dfrac{1}{1+\exp(-0{,}3 \cdot d_{\text{prec}})} - 0{,}5\right] \cdot 2, & \text{si } P \notin [P_{\text{mín}}, P_{\text{máx}}] \\ 0, & \text{si } P \in [P_{\text{mín}}, P_{\text{máx}}] \end{cases} \tag{3.8}$$

donde d_i es la distancia del parámetro i al rango de confort más cercano (inferior o superior). Aquí, iniciamente lo hice utilizando una función lineal de manera que conforme se alejaba del rango óptimo, incrementaba la penalización. Pero esto tenía varios problemas - uno de ellos, por ejemplo, eran que las penalizaciones se hacían muy grandes cuando se alejaba mucho (ocurria tanto esto durante el comienzo del entrenamiento, que el modelo acumulaba muchos rewards negativos y no se recuperaba de ello, señales negativas en exceso). Había que buscar una forma de acotarlo superiormente el reward sin necesidad de forzarlo con las funciones `min/max` de Python, aunque esto servía como *fix temporal*.

No obstante, el mayor problema era que una función lineal no modela algo que me parecía importante: **no es lo mismo estar fuera del rango, pero cerca - que estar fuera y muy lejos**. Recordé las funciones sigmoideas, que ya he mencionado su uso anteriormente; pero durante el desarrollo del proyecto, ésta fue la primera vez que utilicé algo así (en términos cronológicos) - lo cual estaba deseando, porque aunque sean funciones muy utilizadas, pese a haberlas visto y estudiado en múltiples asignaturas, nunca había llegado a poner en práctica. No encaja a la perfección con lo que busco y estoy seguro existe otras alternativas mejores, pero me es más que suficiente y ha terminado funcionando bastante bien. Puedo modular tanto la penalización máxima ajustando el factor de la izquierda (2.0 para temperatura, 1.5 para humedad/precipitación) como la pendiente de la curva con el coeficiente de la distancia. Además, si la distancia es 0, gracias al -0.5 consigo que la penalización sea 0. Veamos la figura 3.25

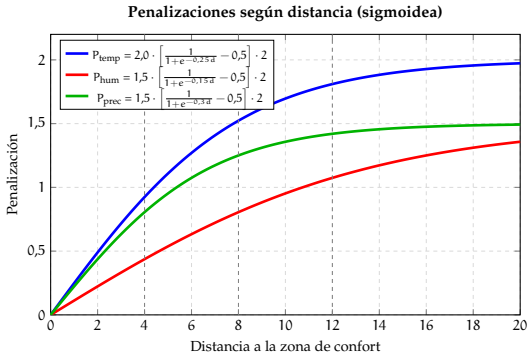


Figura 3.25: Funciones de penalización sigmoideas para las variables climáticas. Aquí se muestra cómo la penalización aumenta de forma suave según la distancia a la zona de confort es mayor. Cada una tiene sus parámetros, esto lo hago para controlar la sensibilidad (con el coeficiente) y la penalización máxima (con factor izda). Los valores son on fruto de la experimentación y observación durante el entrenamiento.

Penalización por valores extremos:

$$P_{extremos} = -\left(P^{ext}_{temp} + P^{ext}_{hum} + P^{ext}_{prec}\right) \tag{3.9}$$

donde:

$$P^{ext}_{temp} = \begin{cases} \text{mín}(3{,}0,\ (T-45)\cdot 0{,}3), & \text{si } T > 50°C \\ \text{mín}(3{,}0,\ (|T|-25)\cdot 0{,}3), & \text{si } T < -25°C \\ 0, & \text{en otro caso} \end{cases} \tag{3.10}$$

$$P^{ext}_{hum} = \begin{cases} \text{mín}(2{,}0,\ (H-95)\cdot 0{,}4), & \text{si } H > 99\,\% \\ \text{mín}(2{,}0,\ (5-H)\cdot 0{,}4), & \text{si } H < 0\,\% \\ 0, & \text{en otro caso} \end{cases} \tag{3.11}$$

$$P^{ext}_{prec} = \begin{cases} \text{mín}(4{,}0,\ (P-(100+P_{máx}))\cdot 0{,}02), & \text{si } P > 100 + P_{máx} \\ 0, & \text{en otro caso} \end{cases} \tag{3.12}$$

La idea es cerciorarnos de que nuestro modelo no pierda en situaciones irrealistas; para ello introduzco una leve penalización que crece de forma suave pero consistente cuando los parámetros toman valores extremos. Así:

- **Temperatura:** A partir de 50 °C o por debajo de 25 °C (es un ejemplo, se ajusta con el mínimo de cualquier bioma), cada grado extra añade una penalización de 0.3 puntos, hasta un tope de 3.0.
- **Humedad:** Para humedades superiores al 99 % o negativas (imposibles), penalizo 0.4 puntos por unidad porcentual de desviación respecto a un rango razonable (5 %–95 %), con un límite de 2.0.
- **Precipitación:** Cuando supera en más de $P_{máx} + 100$ unidades, aplico 0.02 puntos de penalización por unidad extra, hasta 4.0 en total.

Contrasto con respecto a 45, 95, etc, para evitar que empiece a penalizar *justo* al cruzar extremos 50 o 99. Es decir, he querido expresar la noción de **llegando al límite** e intentar transmitir al modelo esa transición hacia la parte negativa (irrealista) de manera gradual y que no empiece directamente en esos valores que considero no deben de darse. *Aviso* al modelo de antemano de que está en zona peligrosa.

La idea de aplicar la penalización de forma más gradual es para **dar al modelo algo de margen para explorar distintas estrategias** antes de ser duro y penalizarle mucho por desviarse. Si se desvía un poco, la penalización es baja y el modelo puede probar libremente diferentes valores - **igual le conviene una penalización pequeña, si en la estrategia que está planificando luego lo compensa al ajustar los 3 parámetros**, por ejemplo. Si doy penalizaciones demasiado grandes, igual *se asusta* y no explora esa idea.
Para terminar, con objeto de poder evitar que el modelo siga aprendiendo incluso en escenarios adversos acoto la recompensa final:

$$R_{final} = \text{máx}\,(-15{,}0, R(s, a)) \tag{3.13}$$

Como reflexión, creo que quizá sería conveniente re-escalar los valores de todos los rewards, ya que ahora mismo **son valores muy grandes los que se consiguen y quizá puede optimizarse**. Por ejemplo, (*Henderson et al., 2018*[47]) hablan sobre sensibilidad extrema de los modelos a la escala de los rewards; afecta mucho a la estabilidad del aprendizaje (si rewards muy grandes o pequeños, pueden saturar gradientes).

3.3.2.6. Entrenamiento

Una vez vistas todas las piezas, veamos el flujo de entrenamiento:

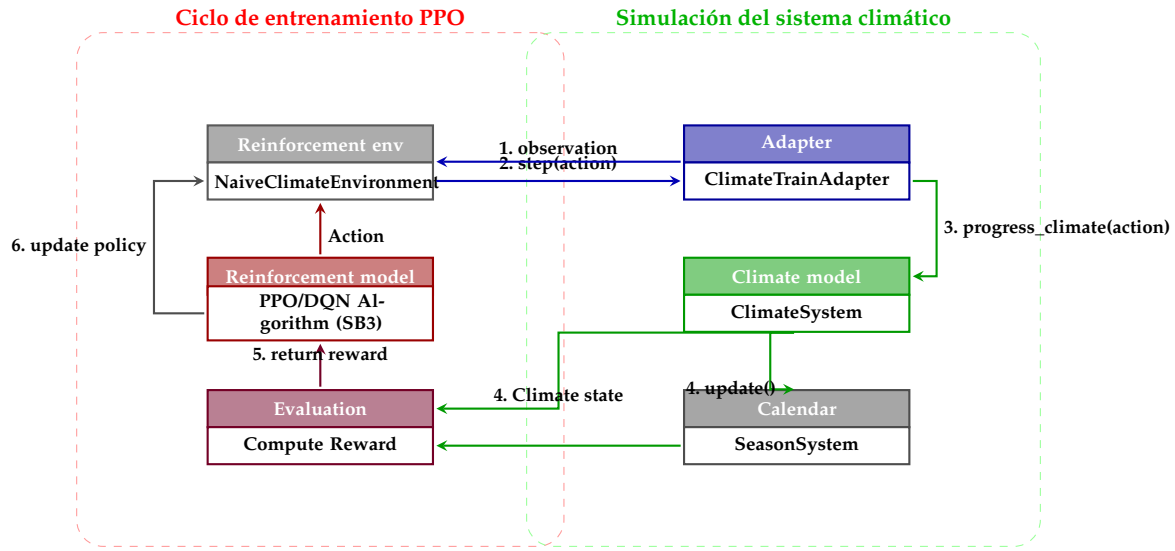


Figura 3.26: Flujo del Adaptador de Clima para entrenamiento con RL

Flujo de entrenamiento

1. **Arranque**. El **Gymnasium Environment** (**NaiveClimateEnvironment**) reinicia el episodio y crea un nuevo **ClimateTrainAdapter** para un bioma concreto. Éste levanta solo los módulos necesarios del bioma: **ClimateSystem** y **SeasonSystem**.
2. **Acción del agente**. El **Algoritmo** (PPO/DQN) envía un índice de *WeatherEvent*; el adaptador lo traduce y ejecuta `progress_climate(action)`, donde `action = index(WeatherEvent)`.
3. **Update de entorno**. El **ClimateSystem** aplica el evento meteorológico al entorno y actualiza el estado climático. Consulta al calendario del **SeasonSystem**.
4. **Obtención del estado**. El adaptador lee el nuevo estado climático.
5. **Evaluación**. Se calcula la recompensa en función del confort climático.
6. **Ajuste de política**. El **algoritmo** recibe la recompensa, actualiza pesos y elige la siguiente acción.
7. **Observación**. El adaptador normaliza temperatura, humedad y precipitación. Tambien recogee estación y tipo de bioma. Devuelve la observación al entorno.

Simulación climática desechable – ¿por qué es un sistema aislado?

Los adaptadores los he preparado para que sean *instancias efímeras*: cada `reset` del **NaiveClimateEnvironment** crea un adaptador nuevo - son de *usar y tirar*; no se arrastran estados entre episodios. Además, el **ClimateTrainAdapter** recibe un tipo de bioma y en base a ello levanta un micro-clima donde se cargan variables atmosféricas y los sistemas de**ClimateSystem + SeasonSystem**; **deja fuera fauna, flora y todas las demás capas del bioma**; y, por último tenemos la *capa de mediación*: ningún otro sistema toca el clima; solo el adaptador tiene esa capacidad.

3.3.2.7. Modelo en inferencia - su integración con la simulación a tiempo real

Una vez entrenado, el agente climático se integra en la simulación principal mediante una clase wrapper con acceso al API de políticas. Primero se carga el modelo preentrenado y, posteriormente, comienza el ciclo:

1. Lo primero es actualizar la estación actual en función del tiempo transcurrido. También se actualizan factores medioambientales si necesario.
2. Después, el agente climático percibe el estado actual y lo normaliza para procesar la observación
3. El modelo de RL decide el evento meteorológico (soleado, lluvia, tormenta, etc.)
4. y se aplican los efectos del evento sobre el estado climático global
5. Notifico a todas las entidades del bioma para que adapten su comportamiento y procesos al nuevo clima
6. Los factores ambientales globales se recalculan ahora considerando factores como la biomasa total, densidad de fauna, niveles de CO_2 o transpiración de la flora

Como podemos observar, **el clima evoluciona de forma dinámica y además responde tanto a ciclos estacionales como a los cambios en el bioma**.

Atmósfera sintética

Veamos unos ejemplos resultantes en diferentes biomas:

Figura 3.27: Climas con diferentes biomas controlados por el agente

Para que esté mejor integrado con el ciclo de retroalimentación y el agente climático responda mejor a cualquier fluctuación dinámica que provenga por cambios en flora, fauna etc, habría que re-entrenarlo ampliando el espacio de observaciones y dándole, de forma controlada variables como la biomasa, densidad de fauna. No obstante, y esto es parte del encanto que tiene - ahora mismo, si existen cambios de manera que la flora baja, sube el CO_2 - al final terminará subiendo la temperatura; el agente climático lanzará eventos meteorológicos no usuales para corregir eso, muchas lluvias, tormentas...por ejemplo. Lo que, pese a ser inverso a como ocurre en la realidad (por el clima - ocurren eventos meteorológicos), el **efecto conseguido con esta vertiente es aproximado y me sirve para simular un clima para cada bioma, convincente y además que se adapta a cambios.**

3.3.3. Agente de fauna

Con este agente se modela el comportamiento de las entidades de fauna - cada organismo tiene que tomar decisiones sobre movimiento, alimentación, respuesta a amenazas, etc, en un entorno dinámico - el bioma que es guiado por la simulación. Como veremos, éste agente sube un escalón en cuanto a complejidad respecto al agente climático. Además, pese a que considero los resultados son decentes - no son tan buenos como en un primer momento me hubiera gustado. No obstante, he disfrutado sobremanera y considero que he aprendido muchísimo también.

3.3.3.1. Justificación del enfoque de RL

Cuando pienso en una entidad con cierto nivel de inteligencia en un entorno 2D (algo que no esté basado simplemente en reglas), directamente se me viene a la cabeza el utilizar Reinforcement Learning. Quizá es por lo visto en la asignatura de ATAI, o quizá por lo sonados que han sido los experimentos que se han hecho con juegos de Atari (*Mnih, Kavukcuoglu, Silver, Graves et al., 2013*[48]), entre muchos otros casos.
He trabajado en el desarrollo de videojuegos y hace años, cuando programábamos sistemas que internamente denominábamos *IA*, no eran más que sistemas de comportamientos modulares o *script behaviours* (StealthAI, PursueAI..etc). Ahora que mi paso por la universidad ha ampliado mi **set de herramientas y capacidades**, quise aprovechar y contemplar otras metodologías, sabiendo que necesito un sistema que me permita que la entidad ejecute acciones en su entorno.

Requisito para el agente de fauna	**RL**	**ML**	**BC**	**MCTS**
Aprender estrategias óptimas a partir de recompensas simples.	✓	✗	✗	✗
Comportamientos emergentes sin reglas explícitas.	✓	✗	✗	✗
No estacionariedad; tiene que adaptarse online a cambios dinámicos en el bioma.	✓	✗	✗	✓
Buena capacidad para espacios de estado/acción grandes y continuos.	✓	✓	✓	✗
Sin dataset etiquetado ni trayectorias expertas.	✓	✗	✗	✓
Objetivos múltiples	✓	✓	✓	✗
Planificación a largo plazo.	✓	✗	✗	✓

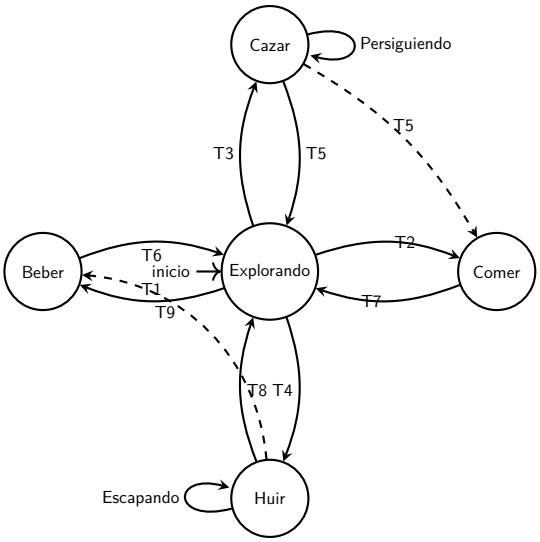
Tabla 3.2: Comparativa de métodos

- **Machine Learning (ML)** - Redes neuronales. Requerirían un enorme set de pares *state–action* ya etiquetados. La idea de generarlo por mi mismo sería inviable o extremadamente costoso, ya que tendría que o bien jugar, o bien programar un agente experto para generar miles de situaciones. Aunque lo hiciera, creo que carecería de capacidad de exploración y se ceñiría a lo visto durante el entrenamiento.
- **Montecarlo Tree Search (MCTS).** Es bueno en cuanto a que no requiere dataset y gestiona bien espacios discretos, pero las constantes recomputaciones del árbol suponen un elevado coste computacional, y más aún en un entorno dinámico como el mio, que no es estacionario.
- **Behavioural Clonning (BC)** Aproveché para investigar un poco sobre esto, pero al poco ví que no me serviria, porque necesita que yo grabe trayectorias óptimas de navegación por el entorno, y mi agente quedaría limitado a imitar ese comportamiento sin capacidad de explorar - como lo mencionado en Machine Learning. Además, cualquier error en navegación lo llevarían a estados desconocidos y no sabría recuperarse. (*Ross et al., 2011*[49])

3.3.3.2. Espacios de observación y acción

Acciones del agente, y sus posibles interacciones con el entorno

El espacio de acción de mi agente de fauna es muy simple, únicamente se consideran **cuantro movimientos direccionales**.
En un principio, también existía la **opción de no desplazarse**, pero ví que no aportaba mucho ya que ciertas interacciones se producen con el cambio a una nueva celda del mapa. Con lo que, ahora mismo siempre estarán en movimiento. No obstante, es posible agregar nuevas acciones de manera relativamente sencilla. Pese a que sean simples, **al interactuar con el entorno emergen comportamientos diferentes**. Veamos a modo de guía conceptual de los comportamientos básicos el siguiente diagrama basado en una máquina de estados finita:



Espacio de acción

- MOVE_NORTH
- MOVE_SOUTH
- MOVE_EAST
- MOVE_WEST

Leyenda

T = Transición
T1: Ver agua + sed
T2: Ver flora + herbívoro
T3: Ver presa + depredador
T4: Ver depredador + ser presa
T5: Ataque exitoso
T6: Sed saciada
T7: Hambre saciada
T8: Escape completado
T9: Agua en ruta

Estado fisiológico e información contextual

A parte de información contextual y, para que el agente tenga toda la información necesaria a la hora de tomar una decisión, he incluido varios factores fisiológicos del agente en la observación. Son factores que determinan el estado y ayudarán al modelo a evaluar la situación:

- **Contexto ambiental**
  - `biome_type`: Índice tipo de bioma $\in \{0, \ldots, |\text{BiomeType}| - 1\}$
  - `diet_type`: Índice tipo de dieta $\in \{0, \ldots, |\text{DietType}| - 1\}$

  Tipos de biomas:
  - 0: tropical
  - 1: desert
  - 2: taiga
  - 3: savanna
  - 4: tundra

- **Estado fisiológico**
  - `thirst_level`: sed $\in [0, 1]$
  - `energy_reserves`: reservas energéticas $\in [0, 1]$
  - `vitality`: vitalidad $\in [0, 1]$
  - `stress_level`: estrés $\in [0, 1]$
  - `hunger_level`: hambre $\in [0, 1]$
  - `somatic_integrity`: integridad somática $\in [0, 1]$

  Tipos de dietas:
  - 0: herbivore
  - 1: carnivore
  - 2: omnivore

3.3.3.3. Percepción espacial

La cognición espacial de los animales en entornos naturales es un tema tremendamente fascinante y complejo; he querido emular la noción de que no perciben su entorno completamente (un león en la savana no ve todo el país), a la hora de actuar sólo lo hacen basándose en información local y limitada, con ello adaptan su comportamiento según lo que encuentran.

En el contexto del proyecto, trabajamos en un entorno 2D, con lo que una aproximación para emular esta visión local puede ser un FoV (Field of View), **para *al menos* intentar reflejar las limitaciones cognitivas de los animales**. Ya había implementado mecanismos similares en juegos en el pasado (Aunque más orientados a LoS (Line of Sight)), pero tenía una referencia marcada de un proyecto

sobre el que trabajamos en la asignatura de ATAI, que utilizaba una técnica similar para que las decisiones sean reactivas basadas en la información local (*Wang et al., 2020*[50])

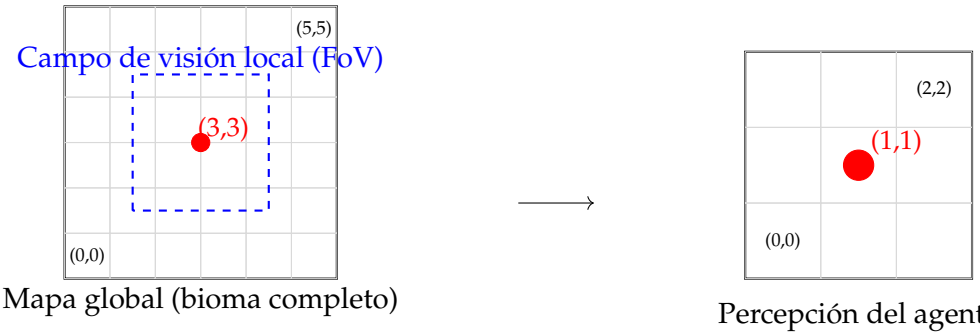


Figura 3.28: Transformación entre coordenadas globales del ecosistema y percepción local

En la figura 3.28 se puede observar el campo de visión del agente **relativo al mapa completo**. Nos interesa únicamente analizar lo que está dentro del recuadro azul y extraer la información que necesitemos, para eso conviene hacer una **transformación o cambio de sistema de referencia** para que el FoV del agente tenga su propio origen - esto nos va a ayudar en el proceso de extracción de información para el mapa local. En este punto, veamos cómo extraigo dos mapas para referencia local, el **mapa del terreno** $\mathcal{L}_{\text{terr}}$ y el **mapa de entidades de flora y fauna** $\mathcal{L}_{\text{ent}}$:

Definiciones iniciales.

Con el `WorldMapManager`, gestiono el mapa del bioma por capas del mismo tamaño, como se explicó en el documento de fundamentos y arquitectura; aquí lo que nos intesa recordar es que existe un capa que hace de mapa con **índices de tipos de terreno**, y otra capa que contiene los **identificadores numéricos** de cada entidad.

Sea (x_g, y_g) la *posición global* del agente en el mapa.

El *mapa global* de terreno es

$$\mathcal{M}_{\text{terr}} \in \mathcal{T}^{H_g \times W_g},$$

donde H_g y W_g son la altura y anchura del mapa global.

El *mapa local* tendrá dimensiones $H_L \times W_L$. Definimos

$$h = \left\lfloor \frac{H_L}{2} \right\rfloor, \quad w = \left\lfloor \frac{W_L}{2} \right\rfloor.$$

Cálculo de los índices del campo de visión en el mapa global:

$$y_{\text{fov}}^{\text{start}} = \text{máx}(0,\ y_g - h), \qquad y_{\text{fov}}^{\text{end}} = \text{mín}(H_g,\ y_g + h + 1),$$
$$x_{\text{fov}}^{\text{start}} = \text{máx}(0,\ x_g - w), \qquad x_{\text{fov}}^{\text{end}} = \text{mín}(W_g,\ x_g + w + 1).$$

Cálculo de los índices correspondientes en el mapa local:

$$y_{\text{loc}}^{\text{start}} = y_{\text{fov}}^{\text{start}} - (y_g - h), \qquad y_{\text{loc}}^{\text{end}} = y_{\text{loc}}^{\text{start}} + (y_{\text{fov}}^{\text{end}} - y_{\text{fov}}^{\text{start}}),$$
$$x_{\text{loc}}^{\text{start}} = x_{\text{fov}}^{\text{start}} - (x_g - w), \qquad x_{\text{loc}}^{\text{end}} = x_{\text{loc}}^{\text{start}} + (x_{\text{fov}}^{\text{end}} - x_{\text{fov}}^{\text{start}}).$$

Inicialización del mapa local.

$$\mathcal{L}_{\text{terr}} = \text{“UNKNOWN”}, \qquad \mathcal{L}_{\text{ent}} = (-1)$$

Inicializo con tipo de terreno `UNKNOWN` - con esto **ya me aseguro de los casos límite donde el agente puede estar en una posición cercana a los bordes del mapa** y el FoV tiene que tener información para entender que es *fuera del mapa*. El mapa de entidades, inicializo a -1.

Rellenado de los mapas locales desde el global:

$$\mathcal{L}_{\text{terr}}\left[y_{\text{loc}}^{\text{start}} : y_{\text{loc}}^{\text{end}},\ x_{\text{loc}}^{\text{start}} : x_{\text{loc}}^{\text{end}}\right] = \mathcal{M}_{\text{terr}}\left[y_{\text{fov}}^{\text{start}} : y_{\text{fov}}^{\text{end}},\ x_{\text{fov}}^{\text{start}} : x_{\text{fov}}^{\text{end}}\right],$$
$$\mathcal{L}_{\text{ent}}\left[y_{\text{loc}}^{\text{start}} : y_{\text{loc}}^{\text{end}},\ x_{\text{loc}}^{\text{start}} : x_{\text{loc}}^{\text{end}}\right] = \mathcal{M}_{\text{ent}}\left[y_{\text{fov}}^{\text{start}} : y_{\text{fov}}^{\text{end}},\ x_{\text{fov}}^{\text{start}} : x_{\text{fov}}^{\text{end}}\right].$$

Con esto, ahora disponemos del **campo perceptivo del agente separado en dos capas de información** - por un lado un mapa local con información de los terrenos de cada casilla y por otro, las entidades. En la implementación lo gestiono con NumPy (Ver código: WorldMapManager - get_local_maps) - esto ahora me facilita la extracción de máscaras especializadas, que me son de enorme utilidad para construir los siguientes **mapas locales al FoV del agente**:

Mapas de percepción

- `terrain_map`: índices del terreno (incluído `UKNOWN`).
- `traversability_map`: máscara binaria con las celdas que son transitables - `WATER_DEEP` y `WATER_MID` son terrenos intransitables. Celdas con obstáculos (otras entidades) se consideran también inválidas.
- `flora_map`: mapa binario de vegetación.
- `prey_map`: mapa binario de presas potenciales para depredadores; especies herbívoras vivas.
- `predator_map`: mapa binario de amenazas, indica posición de otros depredadores.
- `water_map`: mapa binario con celdas de agua poco profunda (`WATER_SHALLOW`).
- `food_map`: un mapa binario de recursos alimenticios: plantas y presas disponibles.

Estos mapas se van a ir actualizando dinámicamente para cada entidad de fauna, en cada paso de la simulación (ya que sólo existirá un solo agente de fauna realmente, como se explicará más adelante). Veamos un ejemplo visual rápido de algunos de los mapas

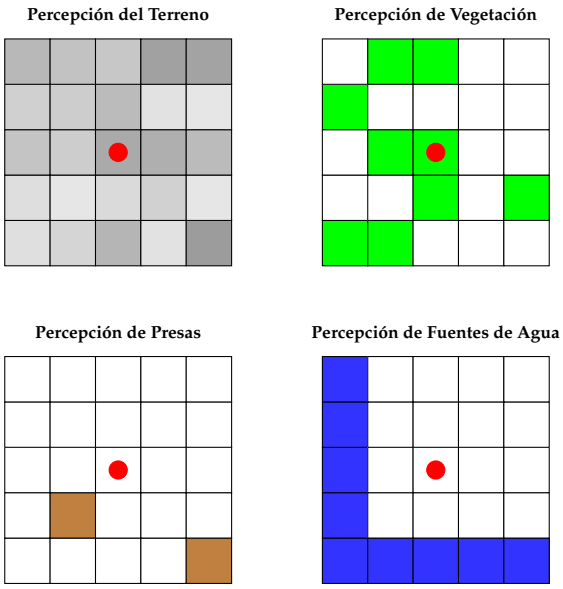
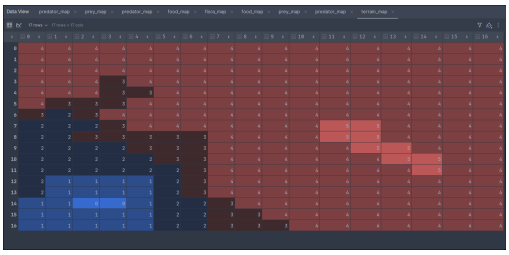


Figura 3.29: Múltiples capas de percepción que forman la representación cognitiva del entorno

Es una mera representación, ya que los mapas realmente tienen este aspecto (en debugger + un plugin para visualizar arrays NumPy como dataframes):

Máscara de terreno en el borde, con `UNKNOWN` (idx 8)



Máscara de terreno

Mapa de depredadores

Figura 3.30: Mapas de activación individuales (FoV: 17x17)

Existe otro mapa que también se considera; **mapa de memoria espacial** que entra en sinergia con el gestor de mapas para así poder proporcionar en cada desplazamiento esta información - con ello se construye un mapa binario con las celdas visitadas. La idea es la de, de manera muy simple, dotar de memoria a los animales al percibir su entorno; por sí solos no son de gran utilidad más que para fomentar exploración, pero en conjunción con el resto de mapas, mi intención es la de proporcionar un modelo *cognitivo* al agente para que pueda sacar relaciones que influyan en decisiones futuras. El mapa se construye así:

$$\text{memoria\_espacial}(x, y) = \begin{cases} 1, & \text{si la posición } (x, y) \text{ ha sido visitada (gestionado por } \texttt{WorldMapManager}) \\ 0, & \text{en caso contrario} \end{cases} \tag{3.14}$$

Claro está, cuando una entidad de fauna se desplaza a una nueva posición, ésta se registra. Con todo esto, nos queda el siguiente **espacio de observación**:

Espacio de observación heterogénea

Mapas espaciales
- Terreno
- Validez
- Visitados
- Flora
- Presas
- Depredadores
- Agua
- Alimento

Estados fisiológico ($\in [0, 1]$)
- Sed ○ Energía ○ Vitalidad
- Estrés ○ Hambre ○ Integridad

Información contextual
- Tipo de bioma
- Dieta

3.3.3.4. Dinámicas de presa-depredador

Antes de mostrar la arquitectura de los modelos y el flujo del entrenamiento, voy a explicar cómo diseñé las dinámicas de presa y depredador.

Para entrenar el agente necesito un entorno aislado y adaptado para Gymnasium - similar al agente climático pero más complejo. El problema que tuve que resolver fue cómo crear un entorno de entrenamiento realista. **Necesitaba un bioma casi completo con fauna, flora, etc.**, pero he aquí la paradoja: quiero dotar de inteligencia al agente, éste es una entidad de fauna que se mueve por el mapa, pero no puede interactuar con el resto de entidades porque están estáticas; sin ningún tipo de *cerebro o inteligencia* - precisamente lo que estaba intentando desarrollar.

Inicialmente lo que se me ocurrió fué entrenar rápidamente un modelo simple que pudiera poner en modo inferencia a todas las entidades excepto al agente en entrenamiento. Pero esta aproximación tenía un problema: si el modelo de las demás entidades contenía errores - que los tenía, era extremadamente básico - el agente terminaría aprendiendo de comportamientos erráticos. Esto no haría más que **amplificar errores** y enseñarle malas prácticas.

Descarté esta idea y opté por crear **dos sistemas** que trabajaran juntos durante el entrenamiento: el primero, `PursuitAndFleeBehaviour`, se encargaría de los comportamientos de persecución y huida.
El segundo vino porque, no me parecia práctico que todas las entidades del mapa estuvieran constantemente huyendo o persiguiendo mientras el agente explora. Así que implementé `LocalInteractionSimulation`, que simula interacciones locales del agente en un radio pequeño (`radius=5` en mis pruebas). Esto es, es un sistema de **activación por proximidad** - sólo las entidades dentro del radio de percepción del agente (similar al FoV) se se activan y adquieren un comportamiento si el agente está cerca:

- **Si el agente es presa**: las entidades cercanas se comportan como depredadores y lo persiguen. El agente debe huir.
- **Si el agente es depredador**: las entidades cercanas actúan como presas y escapan. El agente debe cazar según su interés y capacidad - dependiendo de sus reservas energéticas, hambre, etc.

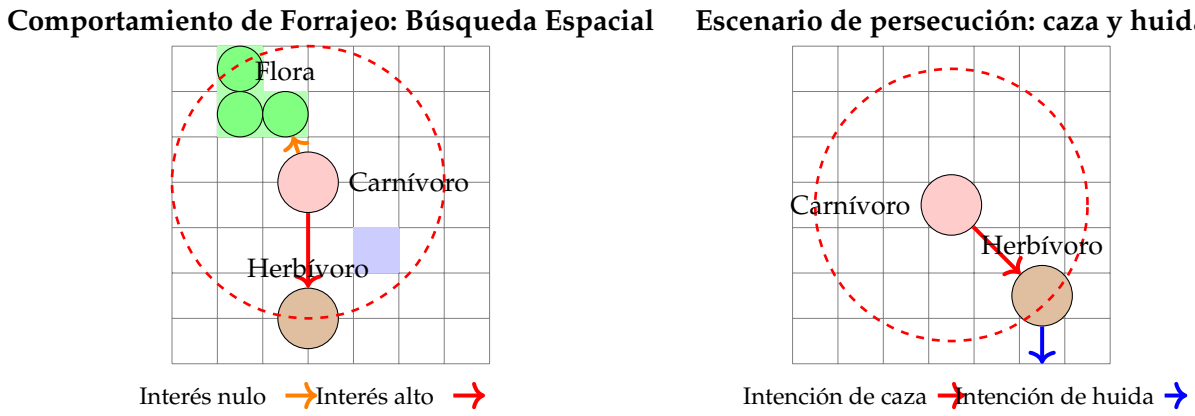


Figura 3.31: Comportamientos de los agentes en el ecosistema simulado

Como se observa en la Figura 3.31 (izquierda), la entidad carnívora detecta tanto entidades (fauna y flora) dentro de su radio de percepción. El algoritmo asigna prioridad a la presa herbívora, ya que los recursos vegetales tienen aporte nulo para el carnívoro. La idea con esta técnica, en sinergia con rewards apropiados (se mostrarán en el apartado correspondiente a rewards), es que **fuerce (o guíe) al agente a aprender éste tipo de dinámicas** e ingeniar estratagemas para conseguirlo. Como nota extra, si una entidad que tenga un comportamiento activo sale del rango del agente, ésta volverá a estar inactiva.

Me ha parecido maravilloso como forzando éstas dinámicas modelamos el comportamiento y, efectivamente; el agente consigue entender lo que tiene que hacer. Veamos de manera esquemática cómo colaboran estos sistemas (3.32).

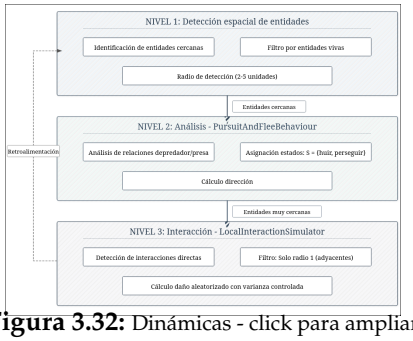


Figura 3.32: Dinámicas - click para ampliar

Al estar en un entorno grid 2D, para el cálculo de la dirección de huída/persecución he seguido unas reglas muy sencillas:

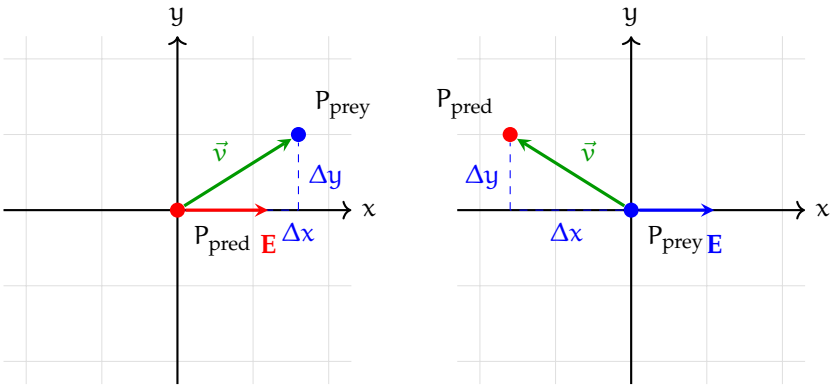


Figura 3.33: Relación depredador-presa: persecución (izqda.) y huida (dcha.).

Paso previo Cálculo del vector desplazamiento:

$$(\Delta x, \Delta y) = \begin{cases} (x_{presa} - x_{dep},\ y_{presa} - y_{dep}), & \textbf{Persecución} \\ (x_{dep} - x_{presa},\ y_{dep} - y_{presa}), & \textbf{Huida} \end{cases}$$

Regla de decisión

$$\text{Dirección} = \begin{cases} \text{Norte} & |\Delta y| > |\Delta x| \wedge \Delta y < 0 \\ \text{Sur} & |\Delta y| > |\Delta x| \wedge \Delta y > 0 \\ \text{Oeste} & |\Delta x| \geq |\Delta y| \wedge \Delta x < 0 \\ \text{Este} & |\Delta x| \geq |\Delta y| \wedge \Delta x > 0 \end{cases}$$

A modo de ejemplo, si tomamos los puntos del gráfico izquierdo:

$$P_{dep} = (0, 0), \quad P_{prey} = (1\acute{6}, 1)$$

1. Desplazamiento:

$$\Delta x = 1{,}6 - 0 = 1{,}6, \quad \Delta y = 1 - 0 = 1$$

2. Dirección cardinal:

$$|\Delta x| = 1{,}6 \geq |\Delta y| = 1, \quad \Delta x > 0 \implies \textbf{Este}$$

Se podría leer como: *Para ir al este, miro si la distancia horizontal es mayor o igual que la vertical, y además si tengo que ir hacia la derecha (diferencia positiva en X).*

3.3.3.5. Modelo de forrajeo

He intentado que el sistema gane realismo cuando la alimentación implica decisiones más allá de la mera proximidad al recurso. Intento reproducir la dinámica trófica con reglas sencillas: cuando una entidad ocupa una casilla adyacente a un recurso, se activa la ingesta. Los herbívoros muerden la planta y obtienen su valor nutritivo, con ello destruyen un % de biomasa pero con una pequeña probabilidad de *arrancarla de raíz*; los carnívoros infligen 15 unidades de daño (reducción de integridad somática) a presas herbívoras y, ocurre algo similar - existe una probabilidad de que los ataques sean letales.

A modo de ejemplo - para beber, la entidad tiene que colocarse sobre una celda con terrreno `WATER_SHALLOW`; y la reposición hídrica se dispara. El cuanto se repone, lo calculo con:

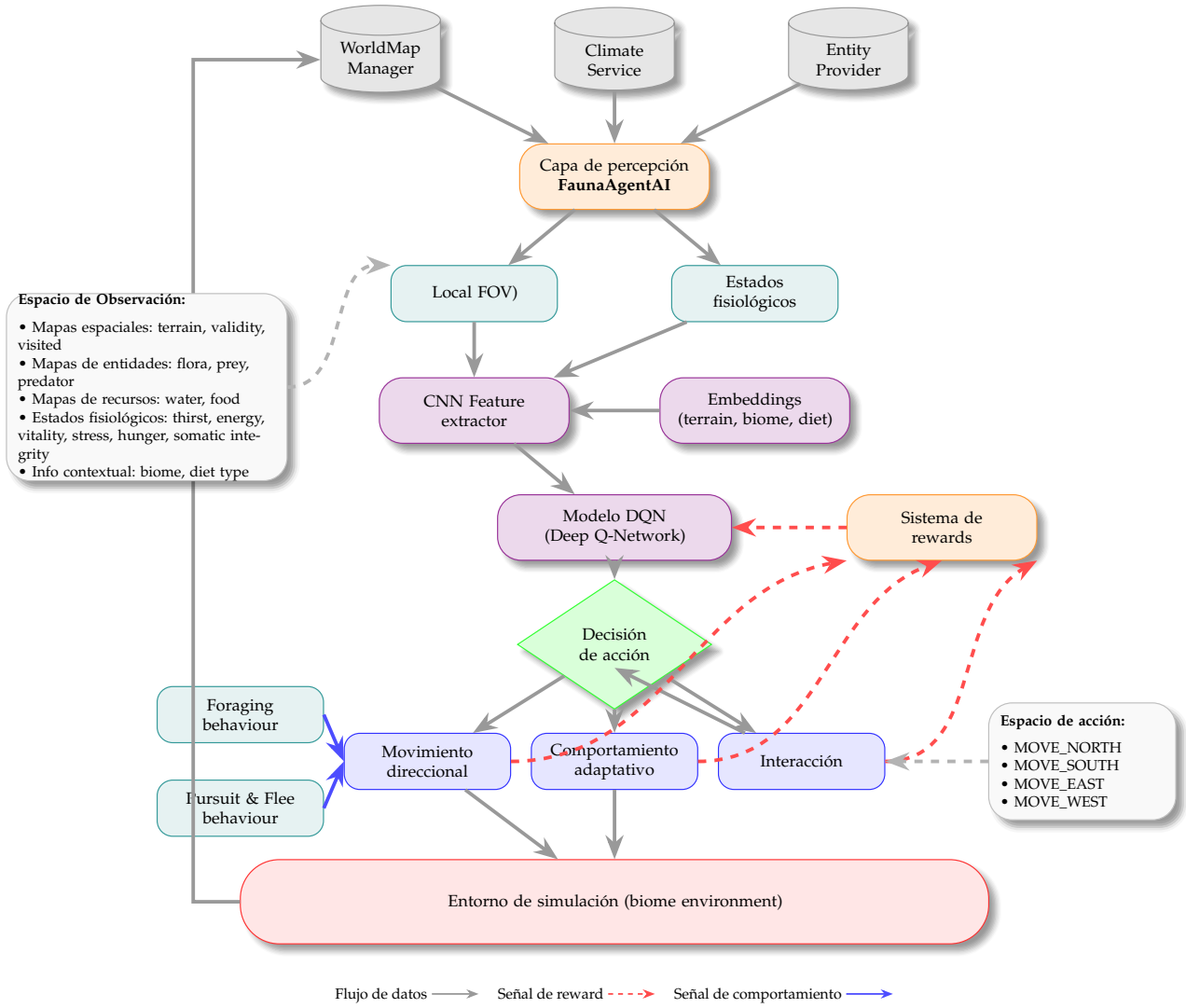
$$H = 5 + (100 - T) \times 0{,}3$$

tal que cuanto mayor es la sed T, mayor es el beneficio H (esto pasa tanto en humanos como en fauna (*E. E. Moore et al., 1987*[51]; *J. A. Roberts et al., 2005*[52])).
En la imagen/gráfico **Apéndice 4.3: Modelo de forrajeo** de pueden consultar el resto de cálculos relacionados al forrajeo.

3.3.3.6. Arquitectura del agente

Antes de proseguir, veamos la arquitectura general y todos sus componentes.

Arquitectura del agente de fauna con deep Reinforcement Learning



Algoritmo

DQN es el algoritmo en el que pensé desde el principio para el agente de fauna. Existen multitud de motivos, pero el que me hacía estar convencido de utilizarlo es precisamente la característica que tiende a representarle: arregla el famoso problema de *moving target* que algoritmos previos de Reinforcement no.

Moving Target y no estacionariedad
Precisamente tratándose de una simulación dinámica donde todo está en constante cambio, necesito de alguna técnica que gestione bien la no estacionariedad. Si bien he experimentado con PPO, me he decantado por DQN porque resuelve mejor este aspecto en entornos cambiantes como el nuestro.

Replay Buffer y aprendizaje
Otra de las cosas que caracteriza a DQN es el **replay buffer**, que en este contexto de terreno 2D con grid, donde genero mapas de biomas proceduralmente, me va a ayudar que el agente aprenda patrones o relaciones esporádicas sin tener de volver a vivirlas en tiempo real.

Por ejemplo, la experiencia de encontrar agua en un bioma de desierto perdura y, teóricamente, enriquecerá la estrategia. También, al romperse correlaciones entre muestras y tener *más datos* (más diversidad con ello), DQN va a converger más rápido, encontrará la política óptima más rápido para cada bioma (se reduce varianza y además se repiten incluso casos poco comunes). A efectos prácticos, si mezcla memorias de diferentes mapas, se va a conseguir una mejor generalización.

3.3.3.7. Extractor de características por defecto de SB3

Como se ha mostrado, el espacio de observación es un **espacio heterogéneo** - dispongo de diferentes tipos de mapas, variables categóricas, continuas, etc. Por ello, al igual que utilicé en el agente climático, hago uso de `MultiInputPolicy` de SB3. Pero con ciertos matices.

Si se indaga sobre qué hace exactamente ésto y como procesa los datos (documentación SB3), podemos observar que al utilizar DQN con `MultiInputPolicy` instancia un extractor de características por cada parte de la observación (utiliza `is_image_space` para diferenciar imagenes) y tras procesar, combina las salidas con su `CombinedExtractor`. ¿Qué implicaciones tiene esto? Veamos:

1. Comportamiento por defecto de SB3 con `MultiInputPolicy`

- **Mapas binarios:** forma $(B, 1, H, W) \rightarrow$ CNN (`NatureCNN`) (aunque también se pueden stackear en todos los mapas en la dimensión de canales (B, C, H, W)).
- **Mapa de terreno:** forma $(B, 1, H, W)$ con valores $\{0, 1, 2, \ldots\} \rightarrow$ misma CNN (tratado como intensidades de píxel).
- **Mapas de índices** (flora, fauna, agua, comida…): forma $(B, 1, H, W) \rightarrow$ mismos filtros convolucionales.
- **Tipo de bioma (`biome_type`):** forma $(B,) \rightarrow$ `one-hot` + `Flatten`.
- **Tipo de dieta (`diet_type`):** forma $(B,) \rightarrow$ `one-hot` + `Flatten`.
- **Escalares** (sed, energía, vitalidad, estrés, hambre, integridad…): forma $(B, 6) \rightarrow$ `Flatten`.
- **Concatenación:** Al final concatena todos los vectores latentes y los pasa al MLP final para calcular los Q-values.

Claro, esto me genera un problema ya que, por un lado tratará los índices de los mapas de terreno, índice de bioma e índice de tipo de dieta como **valores continuos o como intensidades de píxel** (cuando son categóricos) - lo único que puede hacer esto es sacar patrones por contraste. Por otro lado, cada mapa binario será tratado por separado y además, no sabrá que los mapas de terreno se deben de combinar con el resto de mapas para obtener la noción general y sacar patrones de las representaciones conjuntas. Es decir, **la combinación de todos los mapas se debería de pasar de manera conjunta por las mismas capas `CNN`**.

Como se ha mencionado, SB3 convierte las variables categóricas en vectores `one-hot`, esto hace que el modelo no pueda capturar similitudes entre ellas; por eso las paso a un **embedding**: así, biomas afines como tundra y taiga (ambos fríos y con no mucha flora) o tropical y sabana (cálidos y con flora dispersa, por ejemplo) quedan cerca en el espacio vectorial. Por ello, de cara al modelo **la política aprendida en uno sea útil para el otro**, al generalizar mucho mejor, claro.
Por lo explicado y porque - una vez más con espíritu explorador - tenía ganas de hacer mi propio extractor de características, decidí aprovechar la oportunidad que ofrece el API de SB3 para hacer uso de su `policy_kwargs`, y desarrollar dicho extractor para facilitar el aprendizaje a la red.

Además quería sacar el **máximo partido de mis observaciones** - he visto que la intuición inicial de separar los mapas de ésta forma iba bien encaminada; es una estrategia similar a la que utilizó Google DeepMind en Alphastar - su sistema de aprendizaje por refuerzo para bots en StarCraft II -, aunque, claro, la mia, en una versión mucho más sencilla. (*Mathieu et al., 2023*[53])

Sub-espacio de observación	Dtype / shape	¿Lo detecta como imagen?	CombinedExtractor (SB3)	¿Problemas?
8 mapas binarios (*validity_map*, ...)	`Box(0,1,shape=(B,1,H,W), uint8 o float32)`	Sí	`NatureCNN`	Cada mapa se procesa por separado por defecto. Es posible pasar si mismo tipo de datos como (B, C, H, W)
Mapa de índices de terreno (*terrain_map*)	`Box(0,N-1,shape=(B,1,H,W), uint8 o float32)`	Sí	`NatureCNN`	Índices tratados como intensidades de píxel; la CNN capta patrones de contraste en vez de categorías semánticas.
Variables categóricas (*biome_type*, *diet_type*)	`Discrete(n)`	–	`FlattenExtractor + one-hot`	Vectores one-hot de longitud n; sin embedding compacto para capturar similitudes semánticas.
Variables continuas (*thirst_level*, ...)	`Box(shape=(B,6), float32)`	–	`FlattenExtractor`	Correcto: valores reales tratados como escalares.

Tabla 3.3: Tabla resúmen sobre análisis de los espacios de observación e inconvenientes con SB3

2. Patrones espaciales y asociaciones con escalares

Embeddings categóricos:
Mis **índices de terreno**, **bioma** y **dieta** no son ordinales. Con `nn.Embedding` aprendo en un *espacio vectorial continuo* donde *hierba* y *arbusto* pueden quedar más cerca que *agua*. El modelo irá capturando el *significado semántico* de cada embedding en su contexto.

Integración espacio-categorías:
Concateno el `terrain_embedding` con el resto de **canales** y los paso combinados por las **capas convolucionales** – la `CNN` captura *patrones mixtos* gracias a esto; tiene mucha más información y es capaz de entender sinergias entre *mapas de transitabilidad* y el *mapa de terrenos*, por ejemplo.

Integración (continuación):
Lo cual antes, sin extractor customizado no era posible ya que el `CombinedExtractor` de `SB3`, por defecto, haría fueran por separado sin mezclar con el terreno, esto hace que no se puedan aprender **filtros importantes**. No obstante, se pueden hacer modificaciones a la forma del `box` para que trate los *mapas binarios* como **canales**, (B, C, H, W), pero esto seguiría sin incluir los **terrenos**.

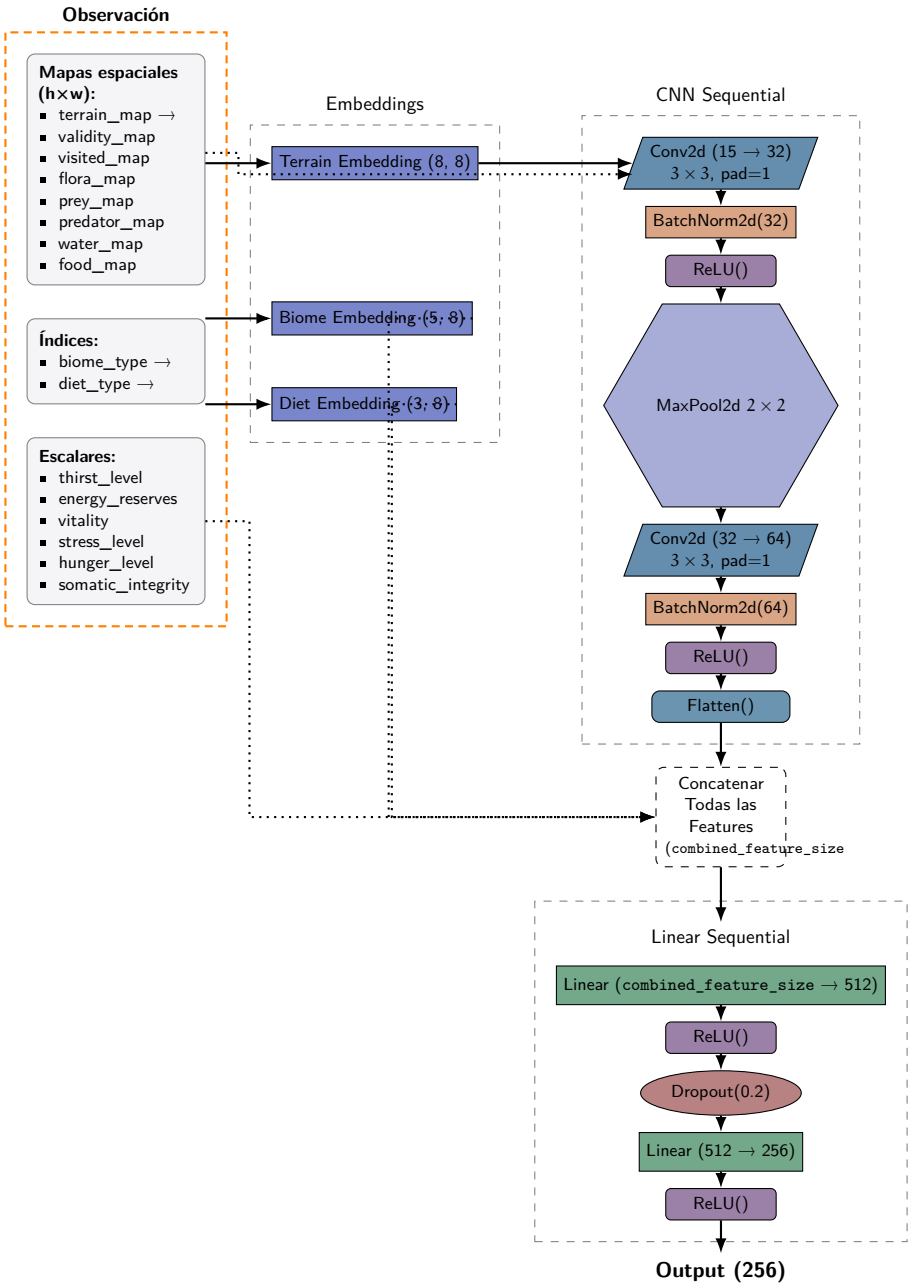
Separación de escalares:
Sed, **energía**, **estrés**, etc., no tienen estructura espacial, así que los aplano (`Flatten`) y concateno tras la parte convolucional.

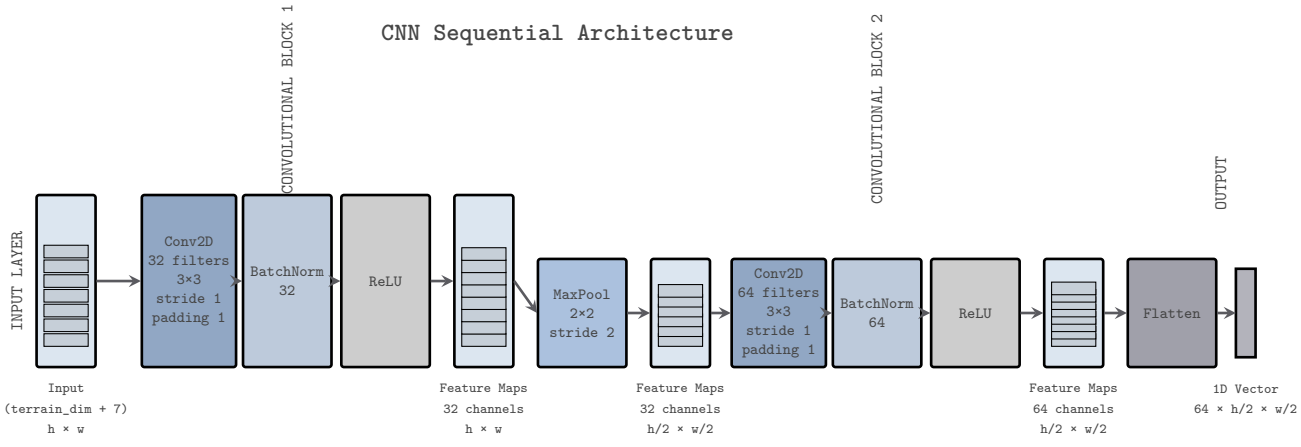
Ya que en SB3 se pueden crear boxes con múltiples mapas (siempre y cuando tengan el mismo tipo de datos) , se me ocurrió que podría transformar los mapas de terrenos: en lugar de un mapa con índices de tipo, distribuirlo de manera que se se tengan tantos mapas binarios como tipos de terreno, donde cada mapa tuviera únicamente a 1 las celdas donde su tipo de terreno exisitera. De ésta manera, le pasaría en (B, C, H, W), C sería $mapas\_previos + mapas\_terrenos$. Podría funcionar, pero decidí mantener la idea de hacer mi propio extractor de características; me parece más elegante, y menor número de canales.

Llevándo todo esto a la parte bonita e idílica; la idea es que el agente pueda percibir su entorno de manera parecida a como lo haría un animal real, que pueda *ver* features espaciales para aprender comportamientos. La red convolucional sería lo más parecido a un sistema sensorial; se encargaría de procesar el campo visual local del agente, y el algoritmo DQN, hace de cerebro que toma decisiones.

Que realmente funcione como se espera, ya es otra cosa.

3.3.3.8. Arquitectura del mi extractor convolucional





Arquitectura

Capa convolucional:

- **Primera convolución:** He hecho algo muy común en este tipo de arquitecturas: pooling seguido de convoluciones que duplican los canales. Empiezo con 32 filtros (3×3, stride 1, padding 1) para mantener la dimensión espacial original (gracias a padding = 1) y generar *features* iniciales que capturan detalles a bajo nivel: patrones simples como transiciones mapas binarios, agrupamientos de agua...etc. Aplico `BatchNorm` para estabilizar entrenamiento y luego `ReLU` para no linealidad.

  Aplico `MaxPool` (2×2, stride 2) que **reduce a la mitad** las dimensiones espaciales, pero mantiene las características más importantes (selecciona el máximo que representa a cada vecindario 2×2). No lo hice en un principio pero, al investigar vi que muchas arquitecturas, como **VGG**, hacían esta combinación de pool + duplicar salida de la siguiente convolución.

- **Segunda convolución:** Justo después, duplico canales a 64 (3×3, stride 1, padding 1). He perdido información con el pooling, pero al duplicar con esta segunda capa los mapas de activación *compenso* - a más neuronas, más combinaciones de patrones complejos que se pueden capturar. No obstante, al ir refinando la representación, ahora detecta zonas de recursos, correlaciones entre mapas, micro-biomas etc. **Nota:** Sacrifico resolución espacial por riqueza en profundidad. *Comprimo, para luego enriquecer expandiendo*. Además, necesitaba reducir cantidad de parámetros (en inferencia la velocidad de simulación se reduce mucho con este modelo).

- **Aplanado final:** Al final, aplano los mapas con `flatten` para producir el vector que condensa toda la información procesada.

Capa lineal:

- **Alternativa considerada:** Me plantee utilizar la técnica de *bottleneck invertido*; descomprimir y luego volver a comprimir, para capturar patrones en espacios vectoriales de mayor dimensión y luego filtrarlos volviendo a reducir dimensión. Pero las combined features son casi siempre de 4166 con lo que si descomprimo tan solo un poco llegaría a $\approx$ 18M parámetros solo en la primera capa; dificil que no me haga overfit. Me parecía excesivo, probé y se notaba mucho en el entrenamiento.

- **Así que decidí** tomar otra perspectiva de reducir progresivamente: *comprimir y comprimir*.

- **Compresión:**

  - Mi vector **flatten tiene toda la información** de las convoluciones.
  - Al comprimirlo a `512` **obligo a destilar información** - fuerzo a las 512 neuronas a que aprendand características útiles del input.
  - Vuelvo a resumir, **lo llevo a un espacio vectorial** de `feature_dims` (256 en training ha sido).
  - Con esto consigo una representación compacta de todo lo procesado por las CNNs, para que el DQN pueda trabajar bien.

He utilizado la técnica de comprimir a 512 ya que DQN Deepmind lo recomendó ((*Mnih, Kavukcuoglu, Silver, Rusu et al., 2015*[54]) aunque depende arquitectura, claro) y es el estándar. Con esto destilo y quito información redundante, luego paso a `features_dim` que es hiperparámetro configurable. No he utilizado más CNNs, para tener un buen equilibrio entre rendimiento y coste computacional, además no son imágenes de alta resolución con lo que es suficiente.

He utilizado únicamente **dos capas de convolución** porque mis mapas de entrada son de baja resolución y relativamente simples a nivel semántico. Para mapas más complejos necesitaría más capas, pero para este caso dos son suficientes.

NOTA IMPORTANTE RELACIONADA: Justo antes entregar, exploré un poco más: **Una última aventura de exploración** (Apéndice, p. 130). Recomiendo consultar esto, después de los resultados (3.41) y no ahora.

Gracias a Tensorboard, Stack Overflow, ChatGPT y alguna que otra cana nueva, conseguí implementar unos callbacks para poder registrar los mapas de activación durante el entrenamiento. Me ha parecido muy interesante poder verlos:

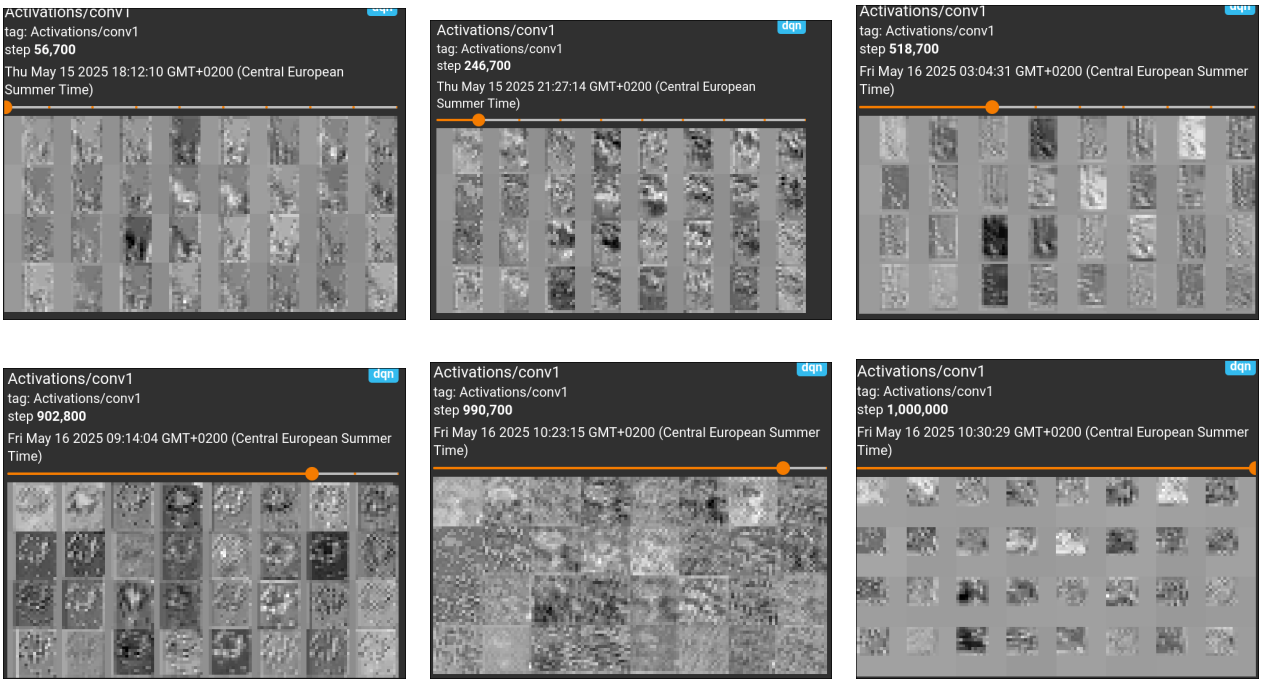


Figura 3.39: Activaciones conv1

Se puede apreciar como a lo largo del entrenamiento, van pasando de un patrón muy denso y con ruido a uno más tenue y mejor definidio. Esto me hace ver que efectivamente, los filtros se van especializando y el extractor va aprendiendo a filtrar lo que le sirve. Al final, termina con ruido.

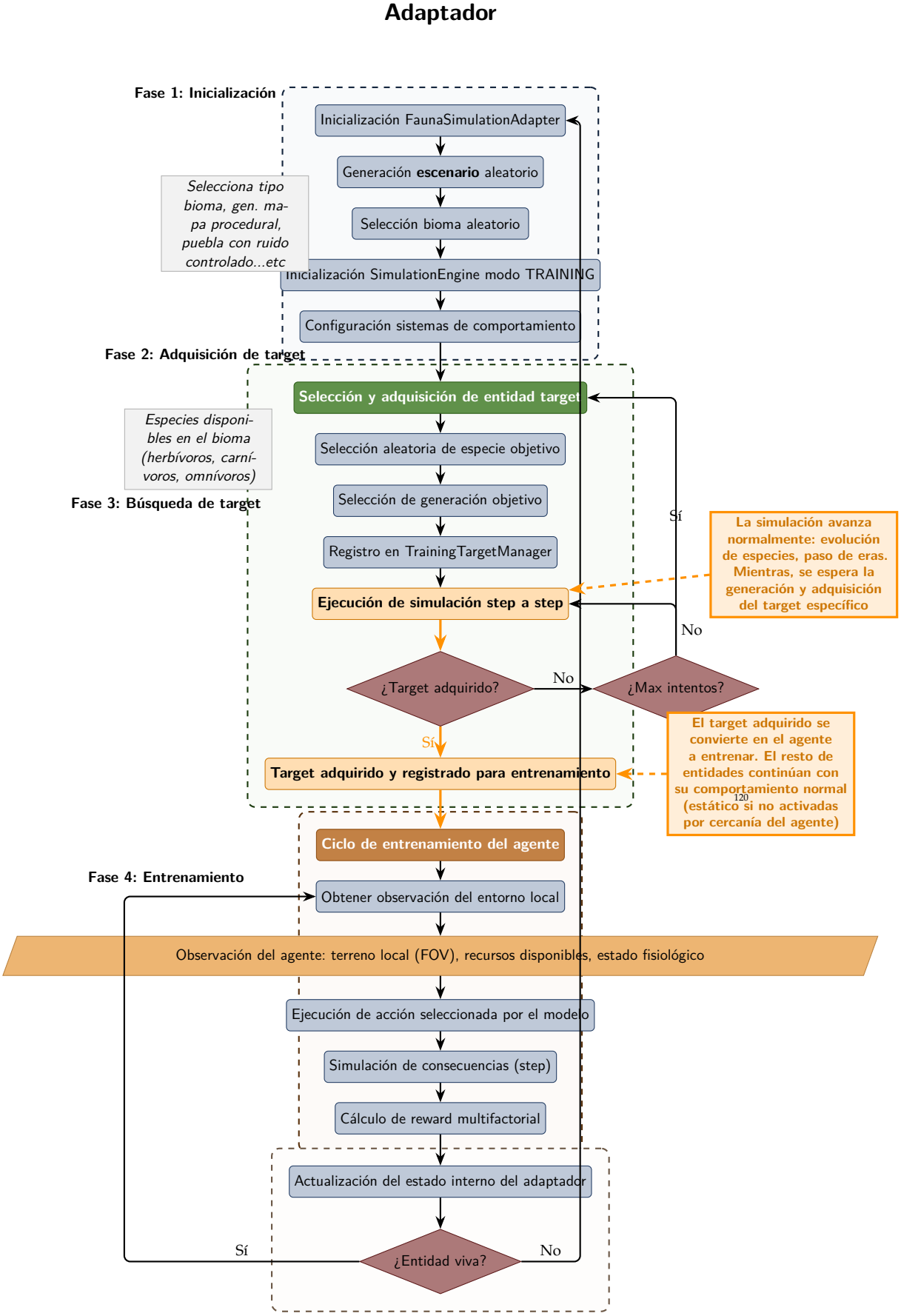
3.3.3.9. Adaptador

Como vimos en el agente de clima, para poder iniciar el entrenamiento tenemos que hacer una implementación del entorno de Gymnasium de SB3, pero esto únicamente nos proporciona una interfaz para interactuar con él, para nosotros poder recibir la acción del modelo y así notificarle el reward que le es otorgado etc. Todo lo demás que sea necesario para gestionar la lógica de la plataforma, requiere de implementación propia.

Con lo que, he implementado una vez más una **arquitectura de adaptación con la que puedo conectar el entorno de Reinforcement de Gymnasium con mis sistemas**. En el caso del adaptador del clima, levantaba únicamente varios módulos a modo de `microclima` y con esos sistemas se hacia un seguimiento - y con ello, se entrenaba el agente.

En este caso es mucho más complejo. El agente representa una entidad de fauna, un animal, que empieza no conociendo nada y aprendiendo según interactúa con su entorno. **Este entorno es un bioma creado por la simulación, con lo que la integración del motor de simulación y bioma, con el entorno de Reinforcement learning, no es trivial.**

Veamos la arquitectura y el flujo de estos sistemas; donde la idea es hacer el agente aprenda estrategias de supervivencia en multitud de entornos ecológicos **totalmente diferentes**, para que aprenda a interactuar en diferentes biomas, situacinoes tróficas, condiciones climáticas etc. **La variedad en la generación de entornos** está pensada para *enseñarle* muchas posibilidades, para intentar que desarrolle comportamientos generalizables.


Adaptador
Fase 1: Inicialización
Inicialización FaunaSimulationAdapter
Generación escenario aleatorio
Selecciona tipo bioma, gen. mapa procedural, puebla con ruido controlado...etc
Selección bioma aleatorio
Inicialización SimulationEngine modo TRAINING
Configuración sistemas de comportamiento
Fase 2: Adquisición de target
Selección y adquisición de entidad target
Especies disponibles en el bioma (herbívoros, carnívoros, omnívoros)
Selección aleatoria de especie objetivo
Fase 3: Búsqueda de target
Selección de generación objetivo
Registro en TrainingTargetManager
Ejecución de simulación step a step
La simulación avanza normalmente: evolución de especies, paso de eras. Mientras, se espera la generación y adquisición del target específico
Sí
No
¿Target adquirido?
No
¿Max intentos?
Sí
Target adquirido y registrado para entrenamiento
El target adquirido se convierte en el agente a entrenar. El resto de entidades continúan con su comportamiento normal (estático si no activadas por cercanía del agente)
Fase 4: Entrenamiento
Ciclo de entrenamiento del agente
Obtener observación del entorno local
Observación del agente: terreno local (FOV), recursos disponibles, estado fisiológico
Ejecución de acción seleccionada por el modelo
Simulación de consecuencias (step)
Cálculo de reward multifactorial
Actualización del estado interno del adaptador
Sí
¿Entidad viva?
No

Fases del proceso de entrenamiento

Fase 1: Inicialización

Cuando mi entorno de Fauna (donde implemento SB3) se inicia, se crea un adaptador y este se encarga, mediante diversas APIs, de generar escenarios aleatorios (pero consistentes) de Biomas. Este escenario está compuesto por diferentes configuraciones, como el tipo de bioma, especies de flora y fauna, y caracteristicas de cada especie.

Fase 2: Adquisición de target

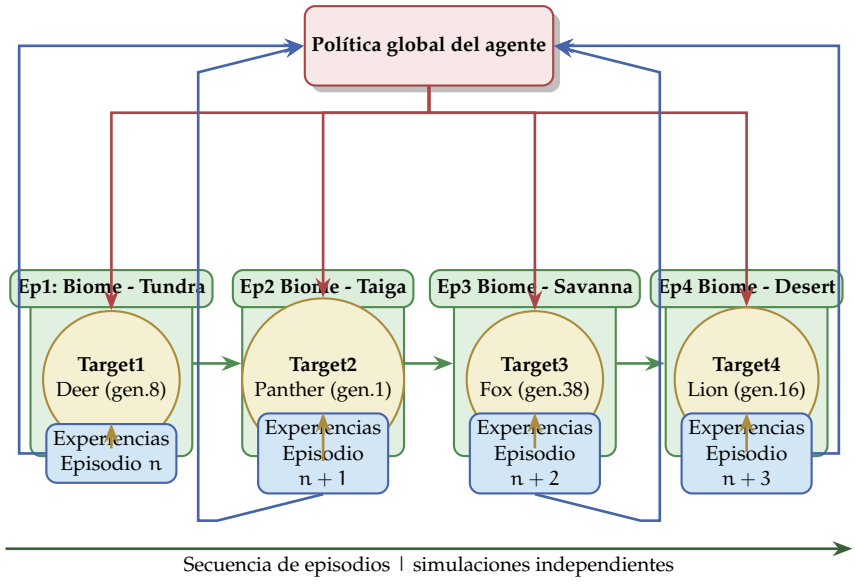
El *target* es una entidad específica de fauna que el adaptador selecciona aleatoriamente para **controlar y entrenar** durante el ciclo actual. Se podría considerar una especie de *agente protagonista*; el que va a interactuar con el ecosistema, tomará decisiones y acumulará experiencias para optimizar la política del modelo. Se me ocurrió este diseño basado en *targets*, pensando una forma tal que el modelo pudiera **aprender de forma progresiva con experiencias de diferentes generaciones evolutivas**.

Una vez el adaptador ha terminado de inicializar y configurar todos los sistemas (comportamientos, forrajeo, persecución/huida, simulador de interacciones locales...etc), se lanza la simulación y la mayoría de sus subsistemas mediante la API correspondiente en modo training (algo más ligero que el modo normal, requiere menos recursos y no utiliza todas las funcionalidades).

Es entonces cuando se **define aleatoriamente un target** - se escogen `(species, entity_type, generation)`, donde especie es una de entre las disponibles en el bioma generado para este episodio y la generación es escogida aleatoriamente (e.g: `(deer, FAUNA, 3)`)

Fase 3: Búsqueda de target

Aquí, la simulación comienza a ejecutarse paso a paso mientras se busca una entidad que cumpla los criterios del **target**. Durante el proceso, el ecosistema evoluciona, pero recordemos que las entidades no exploran y permanecen estáticas. Para llevar a cabo la adjudicación del target - si la generación no es 0 - el agente evolutivo monitoriza el nacimiento de una entidad que cumple esas características y se lo notificará al adaptador. Este target es ahora el agente y se pasa al ciclo de entrenamiento para que interactúe con el entorno y aporte sus experiencas - **en cada episodio, el target/agente contribuye con sus experiencias a la política del modelo**:



Si no se encuentra una entidad target después de un número máximo de intentos, se reinicia el proceso. Si se adquiere exitosamente, la entidad se marca como el target de entrenamiento y se continúa con el ciclo principal.

Fase 4: Entrenamiento

Target adquirido - comienza ciclo entrenamiento

1. Se obtiene la observación del entorno local (`FOV`) desde la perspectiva del target.
2. El agente de *Reinforcement* selecciona una acción basada en las observaciones.
3. La acción se ejecuta en la simulación, el agente se desplaza y se actualiza el Bioma. El adaptador procesa:
   - Simulación local al agente de actividad de las entidades cercanas.
   - Interacciones entre estas entidades.
   - Comportamientos (forrajeo, persecuciones/huidas...etc)
4. Se avanza la simulación un paso temporal y se actualiza el estado interno del adaptador.
5. Se calcula la **señal del reward** basada en:
   - Estado fisiológico del agente
   - Movimientos y exploración
   - Comportamiento de persecución o huida respecto a otras entidades en función de su rol trófico (presa/depredador).
6. Si target no está vivo, se termina el episodio actual; si continúa vivo, se repite el ciclo.

Con todo esto, consigo entrenar con biomas *frescos* y aleatorios constantemente (cada uno con mapas procedurales diferentes), y además con entidades en diferentes momentos de la fase evolutivas y diferentes roles tróficos. Esto hace que las experiencias sean muy variadas con lo que su contribución a la política del agente es muy rica.

Estructura de un episodio

Sabemos que en cada episodio, se crea un nuevo adaptador, con ello se levanta una simulación totalmente nueva e inicia el entrenamiento. Veamos un ejemplo particular rápido:

Estructura de un episodio de entrenamiento para el agente de fauna:

1. **Inicialización del bioma aleatorio**
   Ejemplo simplificado de bioma aleatorio
   - **Tipo**: Tundra (con su mapa procedural, ajuste de componentes de entidades...)
   - **Flora**: `arctic_willow` (14), `tundra_lichen` (26), `arctic_cotton_grass` (22) ...
   - **Fauna**: `polar_bear` (3, carnívoro), `arctic_fox` (5, carnívoro), `musk_ox` (15, herbívoro) ...
2. **Selección del target**
   Simulación en marcha, hasta que nace un target con estas características.
   - **Especie**: `arctic_fox`
   - **Tipo**: FAUNA
   - **Generación**: 3
3. **Observación del entorno**
4. **Ciclo de interacción durante el episodio**
   - Percepción → Decisión → Acción → Reward
   - Comportamientos de persecución/huida
   - Interacciones con otras entidades (solo las cercanas, **simulación por activación**)
   - Forrajeo y alimentación
5. **Terminación del episodio**
   - Por muerte del target
   - Por límite de tiempo

3.3.3.10. Función de reward multicomponente

Función de reward multicomponente

He diseñado esta función con la idea de premiar aquellas decisiones que promuevan supervivencia y penalizar las que ponen en riesgo al agente. Como la del clima; es una función densa, es decir, recibe señal por cada paso. El reward total que recibe el agente lo dictan tres partes: el **estado fisiológico interno, los patrones de movimiento y exploración, y las interacciones con otras entidades**:

$$R_{total} = R_{fisiológico} + R_{movimiento} + R_{comportamiento} \tag{3.15}$$

A continuación muestro cómo cada uno contribuye a la señal general de reward, para un paso en tiempo (timestep) concreto. Si la entidad muere, también existe un caso extra de penalización, se expondrá después de los contribuyentes:

Reward fisiológico

Con la señal que proporciona en conjunto el reward fisiológico, intento expresar la necesidad de mantener la homeostasis interna por parte del agente; esto es, un equilibrio. En este contexto, básicamente se traduce en **gestionar lo mejor posible los recursos vitales** como la energía, la hidratación, saciedad y minimizar el estrés:

$$R_{fisiológico} = R_{base} + \sum_i R_i + \sum_j \Delta R_j \tag{3.16}$$

Donde R_{base} otorga un **pequeño incentivo constante** (0.02) por cada step que el agente sobrevive. R_i son bonificaciones por mantener parámetros en rangos óptimos, y ΔR_j **intenta incentivar la mejora relativa en estos parámetros**; que se comparan con estados previos:

Parámetros básicos:

$$R_{\textbf{energía}} = \begin{cases} +0{,}3 & \text{si energía} > 50\,\% \\ -0{,}3 & \text{si energía} < 15\,\% \end{cases}$$

$$R_{\textbf{hidratación}} = \begin{cases} +0{,}5 & \text{si hidratación} > 40\,\% \\ -0{,}8 & \text{si hidratación} < 20\,\% \end{cases}$$

$$R_{\textbf{saciedad}} = \begin{cases} +0{,}5 & \text{si saciedad} > 30\,\% \\ -0{,}8 & \text{si saciedad} < 20\,\% \end{cases}$$

Parámetros de estado:

$$R_{\textbf{integridad}} = \begin{cases} +0{,}3 & \text{si integridad} > 50\,\% \\ -0{,}3 & \text{si disminuye vs. estado anterior} \end{cases}$$

$$R_{\textbf{vitalidad}} = \begin{cases} +0{,}05 & \text{si vitalidad} > 60\,\% \\ -0{,}1 & \text{si vitalidad} < 20\,\% \end{cases}$$

$$R_{\textbf{estrés}} = \begin{cases} +0{,}05 & \text{si estrés} < 40\,\% \\ -0{,}2 & \text{si estrés} > 80\,\% \end{cases}$$

Mejoras incrementales:

$$\Delta R_{\textbf{hidratación}} = 3{,}0 \times (\text{nivel actual} - \text{nivel anterior})$$
$$\Delta R_{\textbf{saciedad}} = 3{,}0 \times (\text{nivel actual} - \text{nivel anterior})$$
$$\Delta R_{\textbf{energía}} = 2{,}0 \times (\text{nivel actual} - \text{nivel anterior})$$

En el caso de mejoras en el estado de sed, hambre, o reservas energéticas, calculo el reward en función de la magnitud de la mejora: Los factores de 3.0 y 2.0, los he ajustado en función de prioridad relativa y pruebas; también he tenido en cuenta que mejoras en saciedad o hidratación, mejoran pasivamente la energía también. Así que algo menos de importancia les doy.

3.3.3.11. Reward de movimiento

Con este componente, intento modelar la navegación y exploración - quiero que al agente asocie por qué celdas no puede pasar, y a su vez incentivarle a explorar:

$$R_{\text{movimiento}} = \begin{cases} -1{,}0 & \text{si no válido} \\ +0{,}2 & \text{si posición inexplorada} \end{cases} \tag{3.17}$$

Movimientos inválidos son: cuando se intenta traspasar los límites del mapa, intentar posicionarse en una celda que ya está ocupada, o intentar cruzar por terrenos que se han definido como intransitables (`WATER_DEEP` por ejemplo). Por otro lado, le doy un pequeño incentivo (+0.2) para fomentar la exploración - esto ayuda sobre todo al comienzo del entrenamiento para evitar bucles donde repite celdas una y otra vez (igual ha determinado que es mejor solución que otras alternativas).

Este incentivo se lo otorgo únicamente cuando el agente tiene niveles buenos de hidratación y saciedad (superiores al 20 %). La idea es que aprenda prioridades y **en situaciones donde su vida corra peligro, la curiosidad quede a un lado**.

3.3.3.12. Rewards por comportamientos

Por último, con éste reward me centro en premiar las interacciones depredador-presa - la evaluación se adapta al rol del agente:

$$R_{\text{comportamiento}} = \begin{cases} R_{\text{huida}} & \text{si herbivoro (presa)} \\ R_{\text{persecución}} & \text{si carnívoro u omnívoro (depredador)} \end{cases} \tag{3.18}$$

Huída - presa

El reward de huida (cuando el agente sea presa) se calcula para cada depredador cercano y se suma:

$$R_{\text{huida}} = \sum_{\text{depredadores}} R_{\text{huida,d}} \tag{3.19}$$

La idea es, a más depredadores existan en las cercanías, más difícil será escapar. Con lo que **si escapa o al menos intenta escapar** se debería de llevar mejor premio. Veamos ahora el reward individual por cada depredador (en las cercanías):

$$R_{\text{huida,depr}} = \alpha_{\text{dirección}} \cdot f_{\text{amenaza}} \qquad \alpha_{\text{dirección}} = \begin{cases} +0{,}6 & \text{si } \vec{v}_{\text{movimiento}} \cdot \vec{v}_{\text{hacia-depredador}} < 0 \\ -0{,}8 & \text{si } \vec{v}_{\text{movimiento}} \cdot \vec{v}_{\text{hacia-depredador}} \geq 0 \end{cases}$$

Con el factor direccional $\alpha_{\text{dirección}}$ intento determinar si con el movimiento que se acaba de hacer la presa se aleja o acerca del depredador. Utilizo el **dot product** entre el vector de movimiento y el vector que conecta la presa con el depredador: si el resultado es negativo la presa se está alejando (vectores forman ángulo obtuso). Si es positivo, se acerca.

$$f_{\text{amenaza}} = \text{máx}\left(1{,}0, \text{mín}\left(1{,}0, \frac{3{,}0}{d}\right)\right)$$

Con respecto al factor de amenaza f_{amenaza}, es para considerar la proximidad del depredador - me sirve para modular la **urgencia** de la situación y como consecuencia el reward. d es la distancia Manhattan entre agente-depredador.

El efecto que ésta fórmula tiene es: el reward aumenta conforme la distancia disminuye, pero lo acoto para evitar comportamientos *nerviosos* (o abusos por parte del agente) cuando el depredador está lejos.

Caza - depredador

Para el caso cuando el agente sea un depredador, lo que evalúo lo eficaz que sean en localizar presas y acercarse a ellas:

$$R_{\text{persecución}} = \sum_{\text{presas}} R_{\text{persecución,p}} \tag{3.20}$$

Nótese que es un reward acumulado - **nos interesa que el depredador se dirija a la *masa* de presas mayor**, lo que hará que tenga mayor probabilidad de tener éxito. De manera parecida a cómo se premiaba a la presa por huir, premio al depredador por perseguir e intentar cazar, veamos el cálculo que se hace sobre cada presa en las cercanías:

$$R_{\text{persecución,presa}} = \alpha_{\text{acercamiento}} \cdot f_{\text{motivación}} \cdot f_{\text{proximidad}} \tag{3.21}$$

El factor de proximidad $f_{\text{proximidad}}$ me ayuda a regular el reward de manera que cuando la presa está cerca, premia al depredador a cerrar la distancia:

$$f_{\text{proximidad}} = \text{máx}\left(0{,}1, \text{mín}\left(1{,}0, \frac{3{,}0}{d_p}\right)\right) \tag{3.22}$$

Por otro lado, el factor de acercamiento $\alpha_{\text{acercamiento}}$ lo utilizo para ver si el depredador se acerca a la presa, también mediante el dot product:

$$\alpha_{\text{acercamiento}} = \begin{cases} +0{,}6 & \text{si } \vec{v}_{\text{movimiento}} \cdot \vec{v}_{\text{hacia-presa}} > 0 \text{ (se acerca)} \\ -0{,}1 & \text{si } \vec{v}_{\text{movimiento}} \cdot \vec{v}_{\text{hacia-presa}} \leq 0 \text{ y saciedad} > 60\,\% \end{cases} \tag{3.23}$$

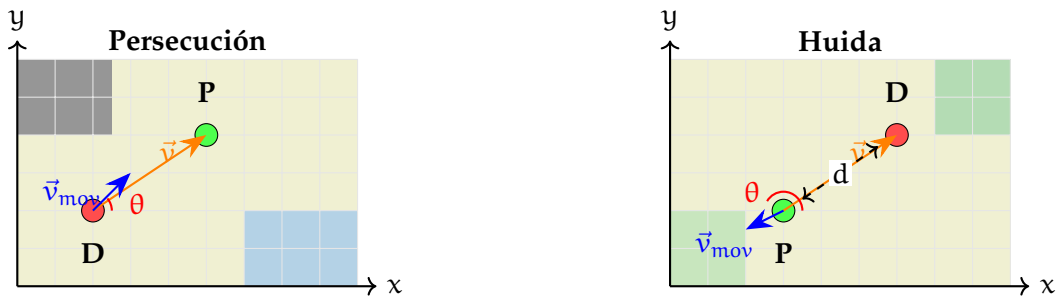
La motivación para cazar $f_{\text{motivación}}$ se modela como una mezcla entre el nivel de saciedad (100 muy saciado - 0 nada) y las reservas energéticas:

$$f_{\text{motivación}} = \left(\frac{100 - \text{nivel de saciedad}}{100}\right) \cdot \left(\frac{\text{reservas de energía}}{\text{reservas máximas}}\right) \tag{3.24}$$

Con esto, dependiendo del nivel de energía que tenga, y también del nivel de hambre, el depredador tendrá más o menos motivación, y eso influirá en el reward final.

Lo que busco es intentar representar una necesidad biológica: un depredador tiene que **equilibrar la necesidad de comer con su capacidad energética para cazar**. Por ejemplo, **un depredador muy hambriento pero sin energía, o con energía pero saciado - va a tener una menor motivación para perseguir**.

Veamos un ejemplo gráfico sencillo de cómo el dot product influye en la decisión - si resultado de hacer el producto punto entre el vector de dirección del origen, y el vector que existe entre el origen y el destino es mayor que 0, implica que se está cercando. No obstante, es un poco simplista, ya que aunque no esté estrictamente yendo en la misma dirección, asumo que, al no estar yendo en la dirección contraria (dot product menor que 0) ya se está acercando. Pero, para este caso, funciona bastante bien.



Penalización por muerte Cuando un agente muere recibe una penalización en función del tiempo vivido - esto es para darle una noción sobre lo importante que es la supervivencia:

$$R_{\text{muerte}} = -15 \cdot \left(1 - \frac{\text{edad al morir}}{\text{lifespan}}\right) \tag{3.25}$$

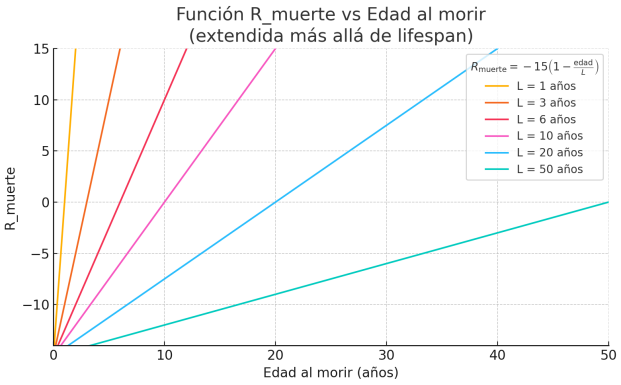


Figura 3.40

Como se puede observar, se traduce en penalizaciones fuertes para muertes prematuras y más leves conforme la muerte se produce más cerca del lifespan o esperanza de vida del agente. Por ejemplo, una entidad que muere al 20 % de su lifespan va a tener una penalización aproximada de -12, y una que alcanza el 80 % de su potencial recibirá solo -3.

> **Excesiva granularidad en la función de rewards**
> Diseñando la función de rewards, me ha resultado complicado no dejarme llevar y modelar pasos inmediatos deseados - esto es, dar rewards al agente por *acabar de hacer algo bien*. Como ya vimos en la asignatura de ATAI, *en un juego de ajedrez, no hay que decir en cada turno qué hacer, si no cómo se gana - ya aprenderá él a jugar*. Con lo que, considero que la función actual de rewards actual debería de ser simplificada y más orientada a que el agente se centre más en su estado fisiológico. No obstante, los resultados han sido decentes, pero me gustaría simplificarla en un futuro.

3.3.3.13. Logs ilustrativos de los ciclos de entrenamiento

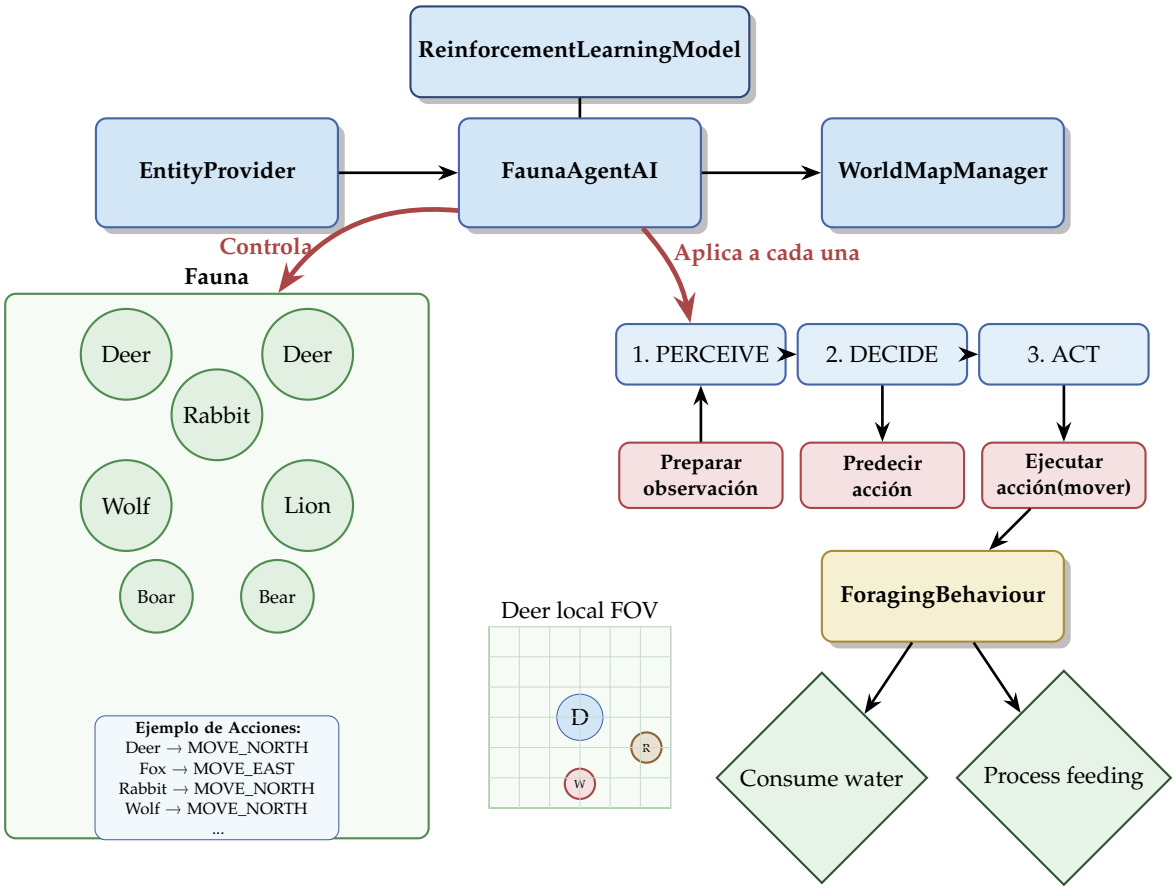
Se puede consultar unos **logs ilustrativos**, con los que muestro el ciclo de entrenamiento explicado, **inicialización de sistemas, búsqueda de target**, etc: **Ciclo de entrenamiento fauna con Reinforcement** (Apéndice, p. 134)

El agente de fauna en inferencia

Con todo esto, **el agente es uno** y dispone de un modelo; con ello dirige a todas las entidades, **decide por cada una de ellas, independientemente de su rol trófico** - ya que **el modelo ha aprendido de ambos y sabe adaptarse a las características que tenga la entidad**. Lo mismo ocurre a los diferentes tipos de bioma, el modelo se adaptará a cada uno.

Una vez generado el modelo con la política óptima, está disponible para ser usado en inferencia. Para ello el agente de fauna carga dicho modelo. Lo más intuitivo quizá sería que cada entidad de fauna fuera considerado un agente, o que al menos cada entidad dispusiera de un modelo para poder *pensar por sí misma* (percibir, decidir y actuar), de manera autónoma - pero un modelo para cada entidad, resulta en una carga enorme y una caída en rendimiento considerable.

Así que el enfoque que le he dado ha sido de de disponer de un agente de fauna global que dirige a todas los los animales del bioma. Para cada una toma su observación particular y en sinergia con el modelo le ayuda a decidir la acción para esa entidad. Cuando la entidad toma la acción, se involucran sistemas de forrajeo como durante el entrenamiento:



3.3.3.14. Entrenamiento

Hyperparámetros

Parámetro	Valor
learning_rate	$5{,}0 \times 10^{-4}$
buffer_size	500 000
learning_starts	10 000
batch_size	64
γ	0.99
τ	0.005
train_freq	4
target_update_interval	5 000
exploration_fraction	0.6
exploration_initial_eps	1.0
exploration_final_eps	0.2
max_grad_norm	1.0
verbose	1
Timesteps: 1 000 000	

Mapa

Tamaño Mapa	35×35
FOV	17

En un principo todas las pruebas las estuve haciendo con mapas mucho más grandes y FOV más pequeños, pero me di cuenta de que para el entrenamiento **mapas más pequeños son suficientes** - en lo que dura un episodio, no es necesario que el agente esté en uno de 100x100, irá experimentando y al poco se creará un mapa nuevo con otra distribución de terreno totalmente nueva. **Lo que aprende en un pequeño con su FOV, le sirve para cualquiera**. Menos recursos, y a efectos prácticos resultará similar. Con respecto al FOV, 7-9 funcionaba bastante bien, pero quería hacer trabajar un poco más al extractor de características y sus CNN, que quería ver su efecto. Mayor coste computacional y más parámetros, pero también representaciones mucho más ricas.

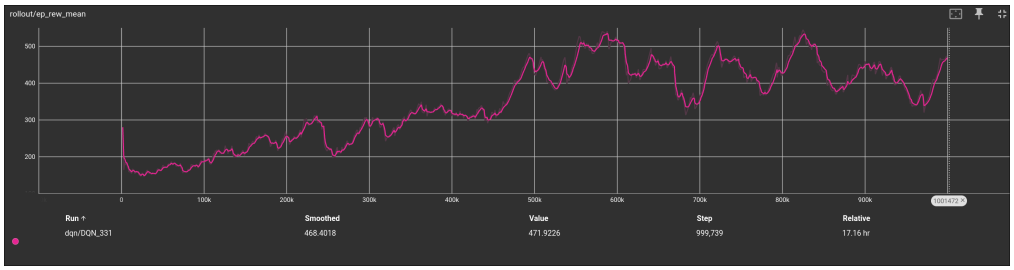


Figura 3.41: Entrenamiento del feature extractor + DQN

Para el entrenamiento del agente de fauna he configurado DQN con hiperparámetros tal que, de nuevo, intenten encontrar ese **punto dulce entre exploración y explotación** - el learning rate de 5×10^{-4} es para intentar tener convergencia gradual sin oscilaciones bruscas, y el buffer de 500.000 experiencias me da una visión muy amplia; **más diversidad** para romper correlaciones temporales. Otro hiperparámetro con el que he jugado bastante con el de exploración, inicialmente empieza en $\epsilon = 1{,}0$ y va decayendo hasta 0,2 durante el 60 % del entrenamiento; con esto consigo que el agente explore bastante antes de empezar a explotar el conocimiento ya adquirido. El $\gamma = 0{,}99$ es para valorar recompensas futuras que, como comenté anteriormente es importante para plantificar a largo plazo.

Como se puede ver en la gráfica, el reward medio evoluciona desde 150 inicialmente hasta estabilizarse alrededor de 450-500 tras 1M de timesteps, con fluctuaciones, pero esto es bastante común en RL. El extractor de características que implementé necesita entrenamiento, con lo que asumo es el motivo por lo que esa primera etapa es un poco más lenta o suave. La tendencia ascendente y posterior estabilidad, me hace ver los hiperparámetros, y en general, todos los sistemas que *wrappean* el entorno de Reinforcement, han hecho su trabajo; siempre mejorable, pero no está mal. Se puede consultar **Comparativa parcial con extractor por defecto de SB3** (Apéndice, p. 130) y **principalmente, las pruebas que hice justo antes de la entrega final Una última aventura de exploración** (Apéndice, p. 130), con resultados muy interesantes.

Implementé unos callbacks para poder obtenter los embeddings aprendidos durante el entrenamiento; de terreno y de bioma, y he utilizado el visor de Projector Tensorflow:

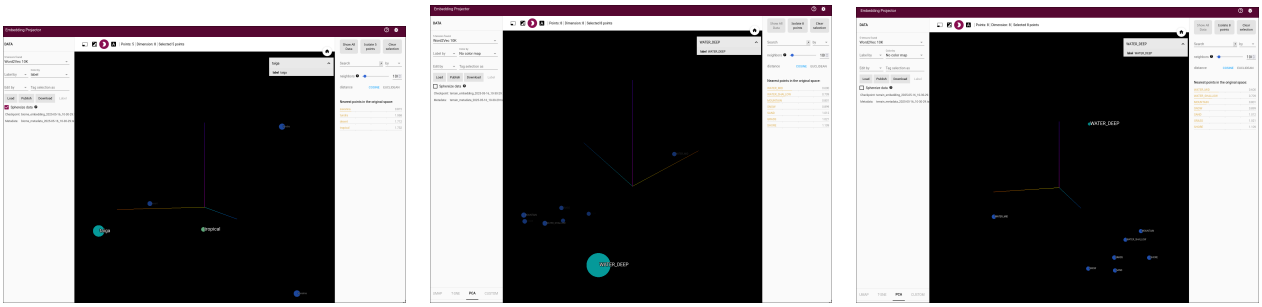


Figura 3.44: Embeddings con PCA

En los tipos de bioma, entiendo que son muchas las cosas que involucran y puede haber encontrado sus relaciones en los espacios abstractos que trabajan los embeddings - pero **en los terrenos me ha sorprendido muy gratamente**; ha separado totalmente los terrenos transitables de los no transitables (`WATER_DEEP, WATER_MID`).

3.3.3.15. Modelo final en Huggingface

Dos modelos preentrenados de Fauna puede encontrarse los siguientes enlaces: fauna-model-200k-fov-17 | | fauna_model_1m

Génesis de inteligencia

(a) Activación de comportamientos

(b) Cervatillo maestro del escape

(c) Estrategias de acorralamiento

(d) Huída por falta de energía

(e) Dispersión de manada + target lock

(f) Consecuencias 0 depredadores

(g) Depredador sobreviviendo

Figura 3.45: Click para ver los videos (con sonido).

En el primer video (a), muestro algunas simulaciones realizadas **para el entrenamiento**, con objeto de ilustrar la activación de comportamientos basado en cercanía; método que he utilizado para que el agente aprenda dinámicas de presa-depredador. **En el resto de videos**, muestro diferentes casuísticas que me han parecido interesantes mostrar en forma de **resultados**, ya en plena simulación. También están disponibles organizados en una playlist. Está lejos de ser perfecto, pero he quedado contento con algunos comportamientos emergentes. **Algunas notas**:

- Para los snapshots, he hecho que se tome 1 snapshot por tick, cuando normalmente es cada mes (60 ticks), con lo que se desplazan 1 casilla cada *medio día*; me sirve para representar *zonas grandes* y ver desplazamientos coherentes en el visor, sin saltos.
- **Debajo de cada video, hay una descripción con los detalles que quiero remarcar de cada escenario**. No obstante, si se tiene algún problema, también las he puesto en el apéndice: **Descripciones de videos de entrenamiento de fauna** (Apéndice, p. 135)
- He intentado **seleccionar a la entidad protagonista** o al menos hacer indicaciones con el cursor del ratón para que quede claro lo que quiero recalcar.
- Si parece que a veces los FPS van lentos, es porque paso manualmente los snapshots *con clicks* y hago una pequeña pausa, para reflejar o intentar que se vea más claro algún movimiento.

3.4. Agente de equilibrium - Inteligencia Artificial Neurosimbólica

3.4.1. Sobre la IA Neurosimbólica

Si se recuerda el *smart population control* que desarrollé en el agente evolutivo, al final comenté que era un sistema que había quedado obsoleto. Mientras que hacer análisis de tendencias poblaciones para balancear la población ha funcionado decentemente, he descubierto otra aproximación que usa métodos más avanzados, que además son lineas de investigación actuales con mucho auge.

Hablo de la Inteligencia Artificial Neurosimbólica, la cual está siendo activamente promovida por organizaciones como IBM Research **IBMNeuroSymbolic**, MIT-IBM Watson AI Lab y DeepMind **Lamb2020** en AlphaGeometry.

Lo que hace esta aproximación es **unir lo mejor de dos mundos** gracias a su arquitectura híbrida; por un lado la **capacidad de aprendizaje de las redes neuronales**, y por otro la **interpretabilidad y razonamiento de los sistemas simbólicos clásicos**. La idea es disponer de una base de conocimiento para dar soporte con reglas explícitas y, combinar los resultados - esto se hace de varias formas:

- **Pipeline clásico (NN → símbolos)**: la red neuronal convierte los datos (eg: etiquetas, embeddings...etc) y el módulo simbólico los recibe como input para razonar; se pasan por la red y luego `if-then`.
- **Late fusion**: los mismos datos (o versiones parecidas) se pasan a ambos módulos; al final, un *meta-juez* combina las salidas de ambos (votación, pesos...). Éste método es el que he utilizado yo.
- **Bucles de realimentación**: las reglas consulta a la red, algo como *¿estás seguro de que hay un semáforo?* y la red re-observa la escena. Es un proceso de ping-pong que aparentemente consigue limpiar ambiguedades y es bastante eficiente computacionalmente porque sólo se reevalúa aquello de lo que haya dudas.

Investigando un poco sobre esto, he llegado a ver algunos métodos muy interesantes, como el NSM (Neural Symbolic Machines) - llegan a mezclar RL con motores simbólicos; entrenan con REINFORCE (policy gradient) un programador neuronal que genera programas en un dialecto de Lisp (*Liang et al., 2017*[55]), y un intérprete simbólico los ejecuta para responder preguntas.
Independientemente de cual se utilice, es gracias al módulo simbólico que se consigue una característica muy mencionada y apreciada en cualquiera de las fuentes que he consultado: la **explicabilidad o interpretabilidad**. El poder explicar claramente la procedencia de una inferencia, a parte de resultar útil por motivos obvios, tiene especial relevancia ya que hoy en día existen leyes que lo exigen.

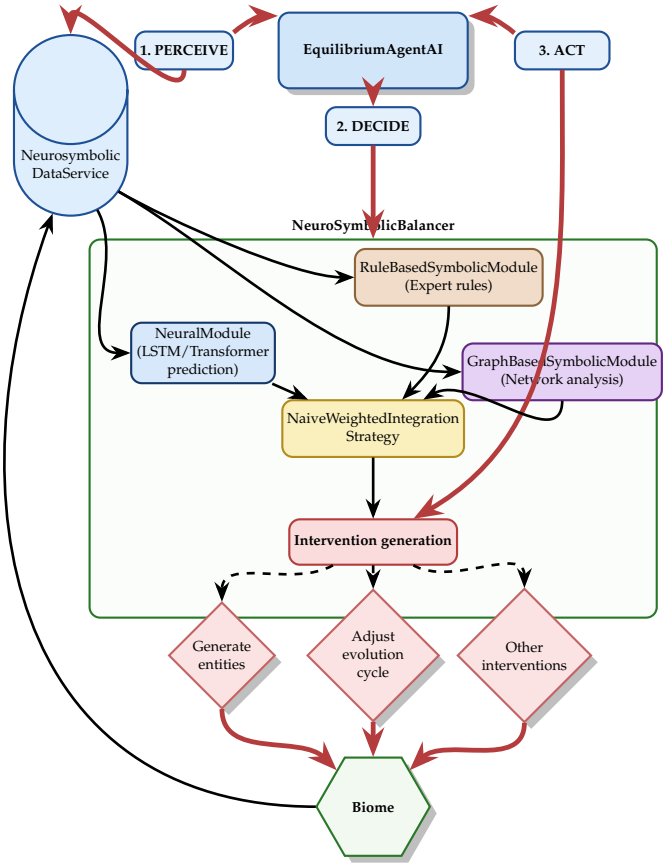
Una de las vertientes más utilizadas para el módulo simbólico son las GNNs. Para este proyecto he preferido explorar los grafos y sus algoritmos como base de conocimiento directamente, ya que, me servía para refrescar y expandir un poco la teoría de grafos, y además es muy utilizada en ecología. **Me ha parecido un método muy interesante que me ha abierto las miras de cara al futuro**

3.4.2. El agente de Equilibrium

Veamos ahora el último de los agentes de Echoes of Gaia: el Agente de *Equilibrium*. Su función es básicamente **supervisar continuamente el ecosistema**; revisa indicios de posibles desequilibrios ecológicos e interviene cuando sea necesario. Esta idea de buscar un balance, la tuve muy al principio, al observar en las primeras simulaciones fluctuaciones y diferentes tipos de desequilibrios que a veces generaban **extinciones en cascada**. Para esto me he inspirado bastante en ecología, por ejemplo en modelos clásicos como el de **Lotka-Volterra** (*Lotka, 1920*[56]; *Volterra, 1926*[57]) , que describen las interacciones depredador-presa o en teoría de redes tróficas (*Cohen et al., 1990*[58]) , que básicamente representan el ecosistema como un grafo dirigido donde se ven claramente las relaciones entre las distintas poblaciones. **Digo inspirado**, porque al menos en el caso de Lotka-Volterra no he trabajado directamente con sus ecuaciones diferenciales, pero sí he tenido en cuenta sus teorías para las dinámicas de presa-depredador y el balance poblacional.

3.4.3. Arquitectura del sistema Neurosimbólico

Como el resto de agentes, sigue un flujo circular de observación, decisión y actuación.



A continuación expongo brevemente los rasgos generales del proceso que es orquestado por el agente **EquilibriumAgentAI**, el cual dispone de un componente especial, el `NeuroSymbolicBalancer`. Como el resto de agentes; implementa el ciclo *cognitivo* de perceive-decide-act y su función es la de intentar mantener el equilibrio del ecosistema. Se irán detallando a lo largo del tema.

NeurosymbolicDataService se comunica con los colectores de datos y métricas para ir produciendo el historial de estados del bioma. Estos estados, son algo parecido a series temporales de datos sobre poblaciones, factores climáticos, métricas, scores, etc., que nos hacen falta para el alimentar a ambos módulos, como se verá posteriormente. Cuando este servicio haya sido alimentado por el recolector de métricas, con tantos snapshots como el número de secuencias determinada en la configuración - **notificará al agente y este comenzará realizando la observación**.

NeuroSymbolicBalancer:

- **NeuralModule**: su núcleo es una red neuronal LSTM (o Transformer, se explicará más sobre esto). Nutriéndose de los datos históricos y habiendo aprendido patrones poblacionales y sobre métricas generales del bioma, intentará predecir tendencias futuras.
- **Módulo Simbólico**: como se ha explicado previamente, analiza el estado actual mediante sistemas lógicos. Se selecciona **una de las dos** estrategias:
  - `RuleBasedSymbolicModule` - **sistema experto clásico**, dominio en relaciones ecológicas.
  - `GraphBasedSymbolicModule` - se analiza la estructura de red del ecosistema + sus propiedades topológicas (ayuda con competencia recursos, presión depredadora etc.)

- **NaiveWeightedIntegrationStrategy**: He utilizado el método de `Late Fusion`, de manera que combino los resultados neurales y simbólicos de forma ponderada.
- **Intervention generation**: Después de la integración, si necesarias, genero intervenciones para mantener el bioma en equilibrio.

No obstante, balancear un sistema con tantas sinergias, procesos, cálculos etc, no es en absoluto trivial. Con el sistema que presentaré, hay una mejora y estabiliza la simulación, pero para que fuera más realista habría que mejorar los tipos de comportamiento - mi vertiente es simplista en comparación a lo que hay por ahí en cuanto a sistemas neurosimbólicos.

3.4.4. Módulo neural: de LSTM a Transformer... y de vuelta a LSTM

3.4.4.1. Origen y estructura de los datos

Para el módulo neural, los datos provienen de dos fuentes diferentes, depende del modo:

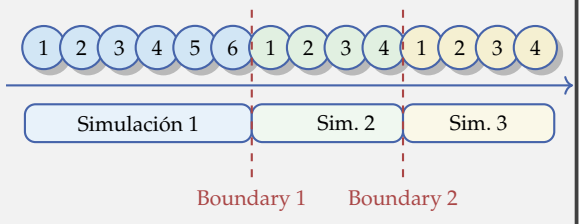
1. **Entrenamiento - genero datasets:** He preparado un sistema específico para la generación de datasets - lanza simulaciones hasta que o bien ocurra una extinción total, o bien pasan las eras que se hayan especificado en la configuración. Esto me produce un fichero `.JSONL` por simulación, y guarda los snapshots que vuelcan el estado del bioma (métricas poblacionales, índices, scores...etc) en un momento concreto. Ver ejemplo de fichero.
2. **Inferencia - series temporales de métricas:** Conforme la simulación avanza, un servicio específico del sistema neurosimbólico va solicitando los datos y estadísticas a los recolectores.

3.4.4.2. Preprocesamiento de datos; ventanas deslizantes en series temporales

Como existe un fichero por simulación, y N **snapshots en cada una**, al juntar todas las simulaciones, hay que tener cuidado con el tratamiento de los datos. Este proceso lo he preparado tal que me permita ofrecer los datos de manera consistente al `DataLoader de PyTorch`, para poder hacer *splits* sin romper la secuencialidad (de las secuencias, valga la redundancia) y no mezclar datos de diferentes simulaciones en una secuencia. Sin el proceso que veremos a continuación, las secuencias estarían mezcladas *a cachos* entre simulaciones y probablemente entre datasets. A grandes rasgos:

1. Detección de límites entre simulaciones

Cuando el ID del snapshot actual es menor que el anterior, marcamos un boundary en ese punto - sirve para no mezclar simulaciones.



2. Segmentación para el split:

Una vez tengo los límites, creo segmentos o rangos que definen el alcance de cada simulación dentro del dataset completo. Por ejemplo, si hay límites en los índices 6 y 10, se crean tres segmentos: (0,6), (6,10) y (10,14). Con estos rangos puedo dividir los datos en datasets de entrenamiento y validación, con esto sé que cada uno tiene simulaciones completas, *sin partir*.

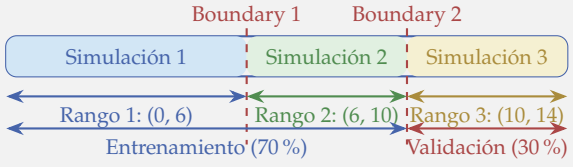


Figura 3.46: Proceso de segmentación y división para entrenamiento-validación

3. Ventanas deslizantes y horizontes: Para ir recorriendo los datos, como es típico en series temporales, lo hago con un sistema de ventanas deslizantes y, además, incluyo un horizonte para los targets / predicciones en inferencia. Durante el proceso se usan tres parámetros:

- **seq_length**: Cuántos pasos temporales uso como input en cada secuencia. Más sobre esto en resultados, pero en general he utilizado 15-25.
- **target_horizon**: Cuántos pasos futuros quiero predecir. En las pruebas, generalmente entre 4-10, aunque he explorado algo más.
- **stride**: Es la distancia entre una ventana y la siguiente; la que mejor me ha funcionado es `seq_length//2` - reduce la redundancia (evita overfitting por secuencias casi idénticas) y consigo que vea más ejemplos sin ser `stride=seq_length` que crearía ventanas disjuntas y perderia información.

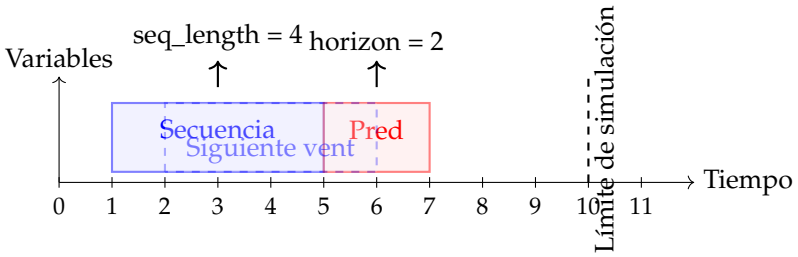


Figura 3.47: Sistema de ventanas deslizantes respetando límites entre simulaciones.

Entonces, en cada paso, se extrae la secuencia de entrada y los targets - además, se detiene antes de cruzar cualquier límite para mantener coherencia. Si no hiciera esto, **no podría representar la evolución del bioma a lo largo del tiempo, habría secuencias inconsistentes**.
Para cada simulación, se hace el siguiente proceso (simplificado):

1. Los datos se dividen utilizando los boundaries.
2. En cada ventana, se utilizan los `seq_length` pasos (secuencias | | snapshots) y los `target_horizon` target features.
3. Con los boundaries que se calcularon, me aseguro que estrictamente **ninguna ventana (`seq + horizon`) cruza el limite de su segmento/simulación**. Si lo hace desecho esta secuencia y sus targets. Pierdo datos, pero gano integridad. Entiendo que seguro existen técnicas para mejorar esto.
4. Avanza la ventana según el `stride`.
5. Se puede ver un ejemplo de división aquí

Un breve ejemplo de cómo se respetan los límites:

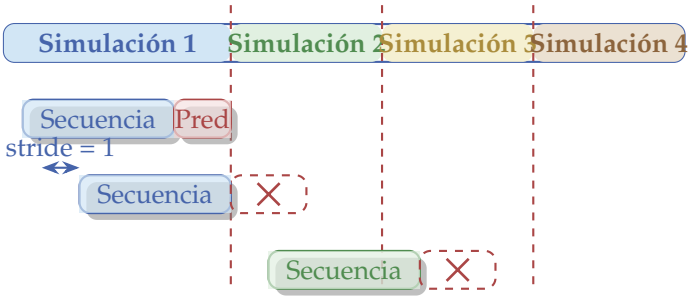


Figura 3.48: Sistema de ventanas deslizantes respetando límites de simulación

Durante el desarrollo, con objeto de poder analizar el funcionamiento de lo que estaba implementando y, ya que a veces el debugger no es suficiente para estas cosas, preparé una herramienta para poder observar la continuidad de las ventanas/targets y observar cómo se iban deslizando correctamente. **Puede verse un ejemplo de un log en Continuidad de ventanas deslizantes** (Apéndice, p. 121)

3.4.4.3. Normalización

Antes de alimentar los datos al modelo, hay que normalizar. Sin comprobar la distribución de mis datos, me lancé a directamente normalizar con `z-score`.
Los resultados no eran buenos - no me había percatado de que **estaba asumiendo que mis datos provenian de una distribución normal**. Pues, lo compruebo - hice un análisis de su distribución:

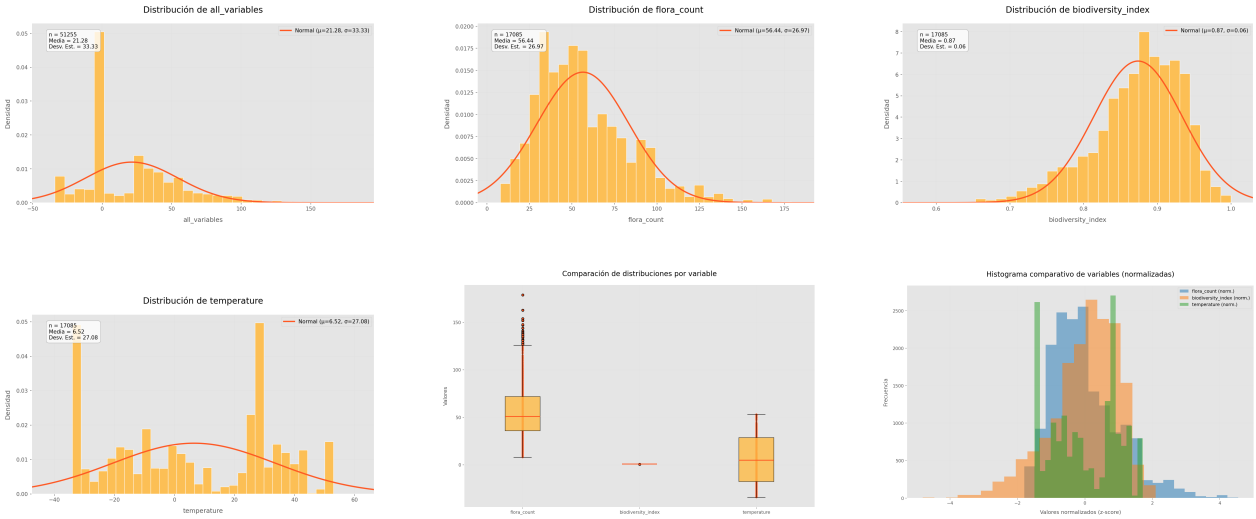


Figura 3.54: Histograma comparativo normalizado

Aquí tenemos unos histogramas de los datos normalizados. Cada bloque que vemos son los intervalos de valores donde la variable aleatoria cae, el eje y es la densidad de observaciones para cada intervalo. También quise dibujar la curva teórica, que parte de una distribución normal ajustada a cada caso (con μ y σ de los datos) - simplemente para tener una referencia visual de como sería si siguiese un modelo normal.

Tabla 3.4: Resumen estadístico de las variables analizadas

Variable	Count	Mean	Std	Skewness	Kurtosis	Shapiro W	p-value	¿Normal?
flora_count	17085	56.44	26.97	0.93	0.85	0.946	1.13e-60	No
biodiversity_index	17085	0.87	0.06	-0.81	0.59	0.960	3.34e-55	No
temperature	17085	6.52	27.08	-0.07	-1.35	0.922	2.92e-68	No

Interpretación de la normalidad

Ninguna de las variables analizadas sigue una distribución normal. He utilizado la prueba Shapiro-Wilk (p-valor $< 0{,}05$ en todos los casos) - utilizo 0.05 por estándar histórico y (*Fisher, 1925*[59]; *Fisher, 1935*[60]).

- **flora_count**: asimetría positiva (0.93), con cola hacia la derecha y acumulación de valores bajos. Por la curtosis positiva (0.85) podemos decir que es una distribución más puntiaguda que la normal.
- **biodiversity_index**: Asimetría negativa (-0.81); concentra valores en la zona alta con cola hacia la izquierda.
- **temperature**: Asimetría casi nula (-0.07); curtosis muy negativa (-1.35) - nos dice que es una distribución aplanada (platicúrtica), es decir; mucho más dispersa que una normal.

Esto me confirma que para la normalización, debería de utilizar métodos estadísticos no paramétricos, ya que **mis datos no siguen una distribución normal**.

Normalización Min-Max Así que probé Min-Max y, para llevar las características al rango [0, 1]:

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Lo aplico a cada característica por separado y los valores mínimos y máximos se calculan del dataset de entrenamiento (como *nota extra*, los serializo en un diccionario junto al modelo, para poder obtenerlos y hacer la inferencia posteriormente).

Con este pequeño cambio **mejoró bastante el rendimiento del modelo**.

3.4.4.4. División de datos

La división de los datos, la he hecho tal que **calculo los estadísticos** (min-max, realmente) para la normalización **exclusivamente del dataset de entrenamiento**. Esto es importante porque si los calculo de todo el dataset, le estoy dando al modelo *pistas* sobre la distribución de validación, y los resultados serían incorrectamente buenos. Además, **si normalizo el dataset de validación con los rangos de entrenamiento**, estoy llevando al *mismo sistema de referencia* los datos de normalización, ergo serán comparables. He dividido los datos de esta manera:

- **Dataset de entrenamiento**: Me quedo con el 70-80 % (depende de la prueba) de los datos para entrenar el modelo.
- **Dataset de validación**: El 20-30 % para validación.

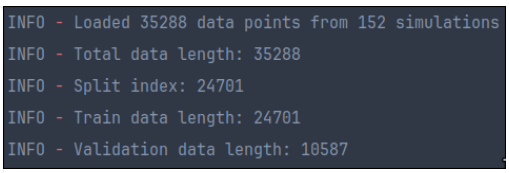


Figura 3.55: Data split

Con todo esto, considero que ya **tengo un flujo de datos que respeta la integridad temporal de las secuencias** y como no dispongo de un volumen enorme de datos, he decidido no crear un conjunto de test adicional. Estos datos, serán con los que se entrenen los modelos que veremos a continuación.

3.4.4.5. De LSTM multivariante...

Para la parte predictiva del agente, empecé con redes LSTM - **en un primer momento** mi idea erea la de predecir el siguiente estado del bioma basándome en los estados previos de una secuencia - siendo estados snapshots.

El dataset con el que partía no es muy grande ($\approx$ 35,000 muestras), con lo que no quería aventurarme con arquitecturas más complejas. Además, ya conocía y me sentía cómodo con la arquitectura básica de las LSTM; tenía algo de experiencia gracias a un proyecto que hice durante la asignatura de Procesamiento de Lenguaje Natural (PLN), el cual disfruté muchísimo, nos dejaron crear el proyecto que quisieramos: LoreNexus. Sin embargo, no profundizaré demasiado en esta versión inicial ya que lo que me interesaba era capturar tendencias más a largo plazo (posibles extinciones o cambios dramáticos), no directamente el siguiente estado. No obstante, como era muy rápido de probar, para contrastar, construí una arquitectura GRU - obtuve resultados muy similares.

Sin adelantar mucho aún - para las pruebas, en este punto utilicé **MSE** como función de pérdida, **estamos ante un problema claramente de regresión**. Profundizaré en estos detalles en el modelo final. Pensando que el problema estaba en la capacidad de la LSTM, decidí explorara arquitecturas más complejas.

3.4.4.6. a Transformers...

Desde hace tiempo tenía ganas de trabajar con transformers. Los descubrí en las asignaturas de PLN y AARN. Leí el famoso paper de **"Attention is All You Need"** (*Vaswani et al., 2017*[61]) y me quedé con ganas de implementarlos en algún proyecto. Me parecía una vertiente un poco pesada para lo que necesitaba, asi que investigué sobre arquitecturas modernas que se pudieran adecuar a mi caso de uso; ví que en los últimos años están apareciendo multitud de formas diferentes orientadas a series temporales, ya que gracias a la atención se pueden modelar distancias mucho más largas que las arquitecturas recurrentes, por ejemplo.

Algunas vertientes de las que encontré **PatchTST** (Yuqi Nie), o **Temporal Convolutional Network (TCN)** (*Bai et al., 2018*[62]), que utiliza convoluciones dilatadas para capturar dependencias a largo plazo sin recurrencia. No obstante, estas arquitecturas están pensadas para horizontes largos y grandes datasets, lo que no terminaba de encajar con mi caso de uso. Aun así, la idea de utilizar horizontes - múltiples pasos futuros para cada variable - me pareció muy acertada, ya que predecir el siguiente valor de una variable que cambie poco, no tiene mucho sentido y **quizá es mejor capturar la tendencia**.

Con lo que pensé en probar con transformers *vanilla* o tradicionales e ir introduciendo, poco a poco elementos que lo reorienten a series temporales - obtuve mejores resultados. He experimentado con distintas configuraciones, pero veamos la arquitectura que mejor me funcionado:

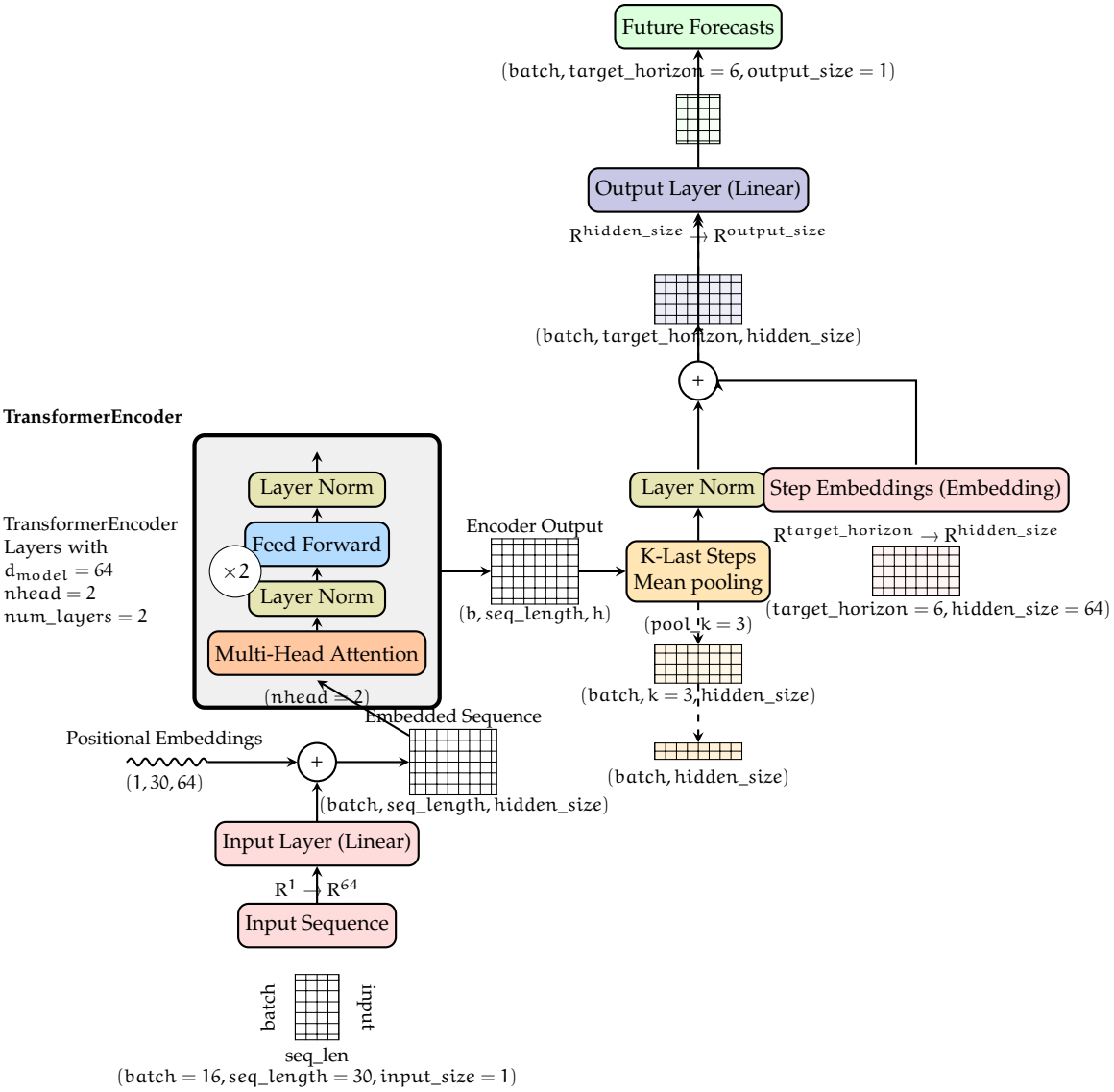


Figura 3.56: Arquitectura de Transformer con multihorizon forecast

Para ésta arquitectura, he pensado en mostrar tensores 2D para una mayor interpretabilidad de qué es cada capa - personalmente, me ayuda mucho el ir indicando la shape en cada paso al implementar el modelo. No se representa, pero muchas operaciones son posibles por diferentes `unsqueezes` y expansiones. El diagrama es meramente informativo.

Arquitectura Transformer para forecasting multihorizon

Input/codificación → agrupación contextual → predicción multihorizonte

Input/Codificación:

- **Embedding lineal inicial:** Incorporo una capa inicial para transformar la dimensionalidad del input y llevarlo a otro espacio vectorial - la idea es que el modelo pueda trabajar con representaciones internas más expresivas. En los *transformers* que he estudiado (la mayoría orientados a PLN), suele haber una primera capa de *embeddings*, pero esto se hace porque al tokenizar identifican el token con un entero para indexar en dicha capa. En mi caso, como trabajo con vectores continuos, una `capa lineal` es suficiente para esta transformación.
- **Embeddings posicionales aprendidos:** He añadido embeddings posicionales *aprendidos* en lugar de los sinusoidales clásicos para introducir información temporal explícita - un Transformer no distingue posiciones relativas por sí solo. El modelo de los sinusoidales es fijo, sin embargo al hacerlos parámetros aprendibles, consigo que se adapten a las posibles irregularidades de mis datos.
- **Capas encoder del Transformer:** Los datos se pasan por varias capas encoder (con 2 capas me ha ido muy bien). Estas capas utilizan mecanismos de *auto-atención* con múltiples cabezales; me sirve para que el modelo extraiga diferentes tipos de relaciones temporales paralelamente. Cada cabezal de atención puede aprender patrones distintos dentro de la secuencia - esto es precisamente una de las mejoras siempre-mencionadas de los transformers al compararlos con métodos como `RNN` o `LSTM`.

Agrupación contextual:

- **Mean-pooling selectivo:** En el *output* del encoder, aplico un **mean-pooling** únicamente sobre los últimos `k` pasos temporales, en vez de usar la secuencia completa. Hice esto porque, según mi intuición, aunque es importante que el modelo analice toda la historia para entender la tendencia general, **los últimos pasos tienen mayor relevancia para la predicción inmediata**. Quise expresar algo parecido a lo que hice con EMA y el clima, en el agente evolutivo. (*Suganthan et al., 2025*[63]; *Azad et al., 2023*[64])
- **Pero, realmente se preserva información:** Al *descartar* el resto de steps de la secuencia salvo los `k` que se hayan configurado, puede parecer que estamos perdiendo toda esa información - pero no es así, el **truco está en la auto-atención**; cada vector del output se ha calculado como una suma ponderada de `todas las representaciones de entrada` (vía las matrices de *queries, keys y values* etc.). Así que, esos `k` steps seleccionados (3 en mi caso) implica que doy mucha más importancia a lo reciente - pero estos a su vez contienen `algo de información` del resto de la secuencia.
- **Normalización de capa:** Después de esto, aplico una **normalización de capa** que ayuda a estabilizar el aprendizaje.

Predicción multihorizonte:

- **Embeddings específicos por horizonte:** Para preparar la *predicción multihorizonte*, añado **embeddings específicos** para cada step o paso futuro del horizonte que quiero predecir. La idea detrás de esto es: no todas las predicciones futuras deben compartir exactamente el mismo contexto. Hago esto para que el modelo aprenda a ajustar la predicción para cada distancia posible del horizonte. (*Guo et al., 2023*[65]; *Nordvik, 2024*[66])
- **Condicionamiento temporal:** Con esto, consigo condicionar el contexto; aplico diferentes *filtros* (los step embeddings) a la misma base (el contexto codificado, los `last-k` steps). Entonces, para cada paso futuro, le sumo un *filtro* específico al contexto, para ajustarlo al target del horizonte que busco predecir - esto es, a la distancia temporal del target. Cada vector de step embedding, se aprende y se ajusta para capturar patrones de su correspondiente target en el horizonte.
- **Proyección final:** Así cada vector de salida final no solo representa información histórica global (de secuencia) y local, sino también la posición *temporal* del target (t = 1, 2, . . . , target_horizon). Finalmente, proyecto esta representación enriquecida al espacio original del problema mediante una **capa lineal**. **Así obtengo directamente los valores previstos para los siguientes pasos temporales**.
- **Inicialización y métricas:** Para la inicialización de pesos, uso Xavier para las capas lineales y una distribución normal estándar en los embeddings. En este punto (cronológico) utilizo MSE como

función de pérdida y, para evaluar el rendimiento, tanto MSE como R^2. **Al no tener una métrica de evaluación tan expresiva como puede ser precisión**, aunque viera que loss bajaba, no tenía una referencia real para saber cuán bien mi modelo estaba rindiendo.

3.4.4.7. Optimizador y Learning rate scheduler

Para el optimizador he utilizado `AdamW` con `OneCycleLR` porque me han funcionado bien y prefería no estar constantemente probando combinaciones. `AdamW` gestiona bien el *weight decay* y `OneCycleLR` me ha dado una progresión del *loss* muy limpia: se nota claramente cómo el *loss* baja siguiendo exactamente el patrón que he configurado en el *scheduler*: sube el *learning rate* hasta el 30 % del entrenamiento y luego hace el decay gradual. He configurado el *scheduler* para que empiece en `max_lr/25` y termine en `max_lr/10000`, y me ha dado convergencia estable en todos los entrenamientos.

3.4.4.8. Baseline

Intentando buscar un punto que me dijese qué tal mi modelo estaba rindiendo de manera algo más precisa, ví que está muy extendido la **comparativa contra modelos baseline**; *modelos simplistas para el caso de uso, que establecen un umbral minimo tal que cualquier modelo más complejo ha de superarlo para justificar su uso*. Con este modelo construido, comparamos nuestro modelo contra esa base, **así disponemos de una referencia**. Contraste por baseline ya había vistro en otras áreas de desarrollo, pero no me hizo *click* hasta que haciendo una especie *crawling* en StackOverflow, Medium y TowardsAI, ví que sería algo que me podría ayudar...

3.4.4.9. Naive forecast: persistenca

Y efectivamente así fué, aprendí sobre los naïve forecasts para predicción de series temporales. Se llama de persistencia, porque parte de la idea es siempre intentar predecir el último valor conocido (también lo llaman no-change). Este modelo nos proporciona unas métricas que nuestro modelo ha de superar - **ahora ya tengo una referencia con la que comparar**, una teoría mucho más solida que únicamente tener MSE. Este es el método que yo he aplicado para construir mi baseline:

$$\hat{y}_{t+h} = y_t \tag{3.26}$$

Donde $\hat{y}_{t+h}$ es la predicción para el horizonte h e y_t es el último valor observado. Esto es lo mismo que asumir que la mejor predicción para cualquier punto futuro es simplemente el último valor conocido:

Yo utilizo una vertiente simple, es decir, al ser series temporales multivariantes, **cada variable tiene tantos targets como target horizon** se haya definido, luego, para cada variable, se pondrá su último valor conocido a todos sus targets.

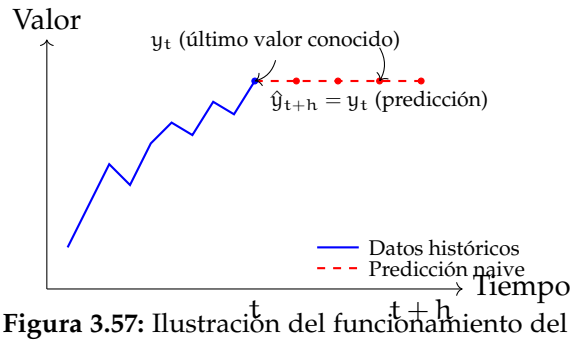
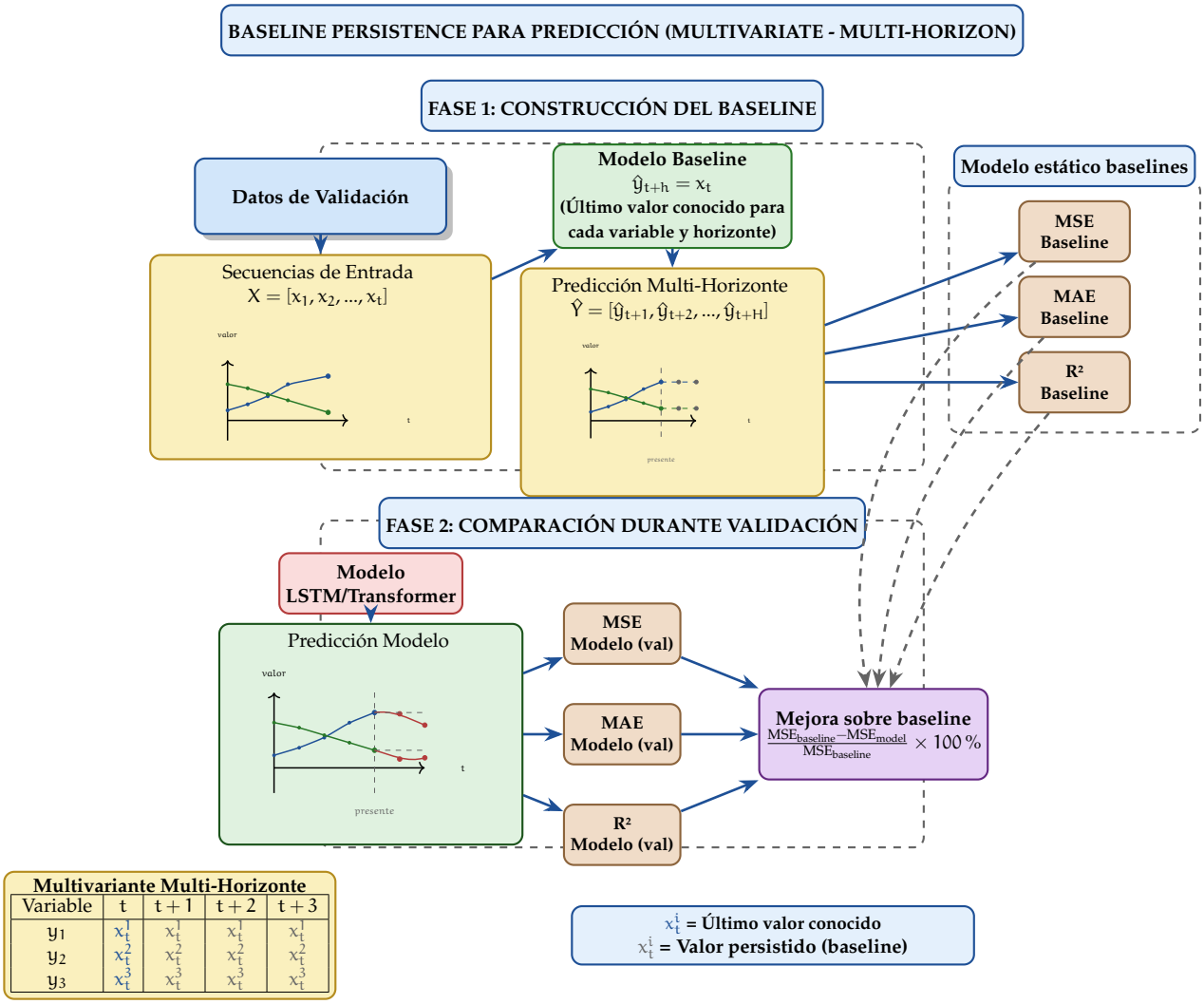


Figura 3.57: Ilustración del funcionamiento del modelo baseline de persistencia: el último valor conocido se proyecta como predicción para todos los horizontes futuros.

3.4.4.10. Limitaciones

Por muy decentes que sean las métricas de error del baseline, no tiene utilidad real para tomar decisiones - es muy limitada (*Hyndman y Athanasopoulos, 2018*[67]); no captura tendencias ni comprende los datos.

3.4.4.11. Construcción del modelo estático de naïve baselines: persistence



Este modelo estático del baseline, se calcula **solo una vez y al inicio del entrenamiento**. En la segunda fase se compara el modelo principal con el baseline - métricas como MSE, MAE y R^2 para ambos. A modo de notas que he ido observando durante el desarrollo, mencionar que:

- **Evaluación exclusiva en conjuntos de validación/test:** El baseline lo construyo con el mismo dataset donde evalúo el modelo - inicialmente lo hacía evaluando en train, hasta que me percaté de que no era lógico. La intuición es - si quiero ser justo, y evalúo mi modelo con el dataset de validación, el baseline debe construirse exactamente con ese mismo conjunto.
- **Experimentación con suavizado:** Probé implementar una variante del baseline con Medias Móviles Exponenciales (EMA) para reducir el ruido. Pero en los resultados se veía que, aunque bajaba la varianza, introducía un sesgo bastante grande y descontrolaba el baseline. Al final mantuve el baseline de persistencia puro, sin suavizado.
- **Horizonte múltiple:** Con objeto de ver cómo empeora la precisión cuanto más lejos intentamos predecir, probé el rendimiento a diferentes distancias. En el baseline esto se nota muchísimo, la precisión cae en picado con horizontes largos.
- **Múltiples métricas:** Utilicé tanto MAE (Error Absoluto Medio) como RMSE (Error Cuadrático Medio). Detallaré esto en más profundidad en el modelo final.

3.4.4.12. Rendimiento del modelo de persistencia

En cualquier experimento de los que he hecho, el rendimiento de *baselines* es sorprendentemente bueno. Al principio, aunque fuera lógico quise revisar literatura al respecto y encontré Deforce et al. (2024), donde afirman:

> *«Consequently, the relatively strong performance of the naive model can be attributed to the fact that ψ_{soil} remains relatively close to the last known value.»*

BASELINE METRICS	
MSE	0.004040
MAE	0.033892
R^2	0.704300

Como se mencionó anteriormente, por muy decentes que sean las métricas, su utilidad real para tomar decisiones es baja o nula. (*Deforce et al., 2024*[68]) lo explican bien: *«However, such naive forecasts are not useful to the practitioner as they do not contain any new information»*.

También, lo que he comprobado es que utilizar un horizonte de predicción largo ($h = 10$) me sirve para ver realmente si el modelo aprende o no. En predicciones a corto plazo, el *baseline* de persistencia es sorprendentemente bueno. Sin embargo, en horizontes largos es donde se nota la diferencia, y si mi modelo tiende a hacerlo bastane mejor, ya tengo una referencia buena para ver si realmente me aprende.

Tiene sentido, ya que en horizontes cortos la autocorrelación de las series hace que la persistencia funcione bien, pero cuando intentamos predecir más lejos, esta autocorrelación se pierde, y las relaciones no lineales entre variables se empiezan a notar.

3.4.4.13. Función de pérdida mejorada gracias al baseline

Solo por lo que voy a exponer en esta parte, me ha encantado aplicar el baselines; al igual que con el modelo LSTM, en este punto aún utilizaba MSE como función de pérdida, y para evaluar el rendimiento, tanto MSE como R2. Al no tener una métrica de evaluación tan expresiva como puede ser precisión, **aunque viera que loss bajaba**, no tenía una referencia real para saber cuán bien mi modelo estaba rindiendo. El MSE que yo veo, puede parecer bajo y realmente no serlo.

Pese a que el MSE del modelo era mejor que el del baselines, gracias a estas comparativas pude observar que el MAE era peor - sin esta comparativa, no tenía una referencia para ver que mi MAE necesitaba mejorar urgentemente. No obstante, es algo normal asumo, ya que en ésta vertiente el baseline tendrá error absoluto bajo al tener deltas pequeñitos en bastantes variables. Un ejemplo de resultados de 3 experimentos:

Tabla 3.5: Modelo transformer - función de pérdida: `MSELoss()` (configuraciones)

Exp	Métrica	Modelo	Baseline	Mejora (%)
1	MSE	0.0018	0.0024	19.46
	MAE	0.0267	0.0231	-15.36
	R^2	0.8609	0.8230	–
2	MSE	0.0029	0.0039	25.44
	MAE	0.0351	0.0335	-4.98
	R^2	0.7751	0.7012	–
3	MSE	0.0029	0.0039	24.52
	MAE	0.0348	0.0335	-3.96
	R^2	0.7780	0.7012	–

	Hiperparámetros			
Exp	Parámetro - Valor		Parámetro - Valor	
1	lr	0.001	batch_size	32
	dropout	0.4	hidden_size	64
	num_layers	2	weight_decay	0.0001
	target_horizon	5	sequence_length	15
2	lr	0.001	batch_size	32
	dropout	0.1	hidden_size	64
	num_layers	2	weight_decay	0.001
	target_horizon	10	sequence_length	15
3	lr	0.001	batch_size	32
	dropout	0.2	hidden_size	64
	num_layers	3	weight_decay	0.001
	target_horizon	10	sequence_length	15

Entonces, esto me hizo decidirme por introducir `MAE`, tal que utilicé `MSE` + `MAE` ponderadas como **función de pérdida compuesta**; los resultados empezaron a mejorar. Ajusté los pesos manualmente y noté cómo encontraba un balance, con **0.7 para `MSE` y 0.3 para`MAE`** como **mejor combinación**.

Pequeño paréntesis: intento con pesos adaptativos

Intentando mejorar métricas e investigando sobre funciones de pérdida compuestas ponderadas, encontré un paper que me animé a probar sobre pesos adaptativos para la función de pérdida. La idea está basada en (*Kendall et al., 2018*[69]), que proponen aprender automáticamente cuánto peso dar a cada componente de la pérdida según su incertidumbre. Aquí ya no se pondera manualmente, si no que el modelo aprende parámetros σ_i para cada término y los ajusta:

$$\tilde{L}_i = \frac{1}{2\sigma_i^2} L_i + \log \sigma_i \tag{3.27}$$

La implementación la hice añadiendo estos σ_i como parámetros del modelo simplemente, pero me encontré con un par de problemas que no conseguí solucionar: a veces la pérdida convergía a valores negativos (lo cual no tiene sentido) y algunos parámetros σ_i se disparaban. No encontré nada sobre estas pérdidas negativas, así que acabé desistiendo. No obstante, me quedo con la idea para el futuro, tiene potencia para ser muy útil.

Pero, luego descubrí `SmoothL1Loss` – que hace lo que yo estaba buscando, pero mejor que como yo lo había expresado. Esto me llevó inevitablemente a seguir investigando para mejorar, hasta que di con `Huber Loss`, con la cual me he quedado **definitivamente** ya que es la que mejor me ha funcionado. Sistemáticamente, ambas hacen lo mismo, pero se diferencian en implementaciones y parametrización internas.

Lo interesante de `Huber Loss` para mi caso, es que ofrece las ventajas de ambas: `MSE` (buena para errores pequeños) y `MAE` (protección contra outliers). Probando, he terminado utilizando **Huber + MAE ponderadas**. Pese a que Huber sea adaptativa y trate errores grandes, al haber tenido mejores resultados agregando `MAE`, infiero que estoy suavizando y protegiendo aún más contra *outliers*. Encaja muy bien con mis datos - que mezclan pequeños cambios en algunas variables y fluctuaciones grandes en otras.

Se puede ver unas gráficas sobre la continuidad de la ventana actual y el target multihorizonte a predecir; se puede apreciar cómo al ser multivariante es algo complejo, y cómo a más largo el horizonte - más incierto es el futuro y complicado de modelar.**Continuidad en ventana vs salto en horizonte de una característica** (Apéndice, p. 123).

3.4.4.14. Con la nueva función de pérdida y el baselines

Tabla 3.6: Resultados del modelo transformer - función de pérdida: `MSELoss()`

Exp	Métrica	Modelo	Baseline	Mejora (%)
1	MSE	0.0017	0.0024	28.13
	MAE	0.0229	0.0231	0.87
	R^2	0.8795	0.8230	–
2	MSE	0.0027	0.0039	29.62
	MAE	0.0319	0.0335	4.73
	R^2	0.7965	0.7012	–
3	MSE	0.0029	0.0039	25.54
	MAE	0.0335	0.0335	-0.06
	R^2	0.7833	0.7012	–

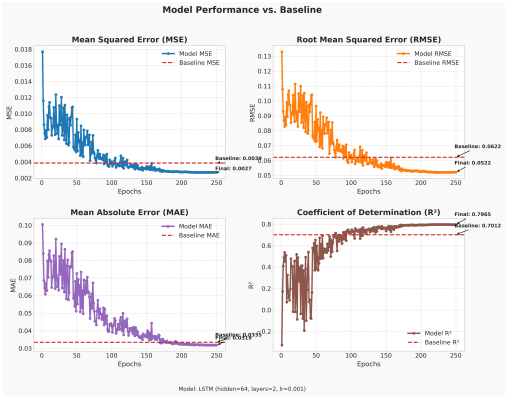


Figura 3.58: Evolución de todas las métricas durante el entrenamiento

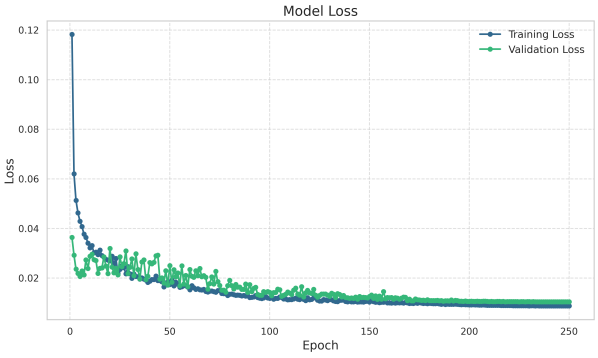


Figura 3.59: Pérdida de entrenamiento y validación - rápida convergencia.

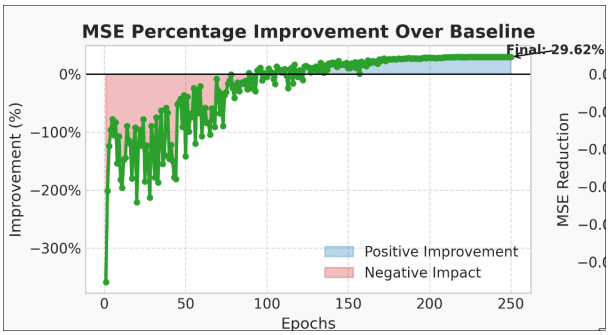


Figura 3.60: Exp 2. Mejora MSE sobre baseline $\approx$ 30 %.

La diferencia es decente, pero **lo que me llevo de esto es el proceso**, ya que gracias a tener un baseline como referencia, me he centrado en mejorar el MAE, y como consecuencia el MSE también ha mejorado. En los 3 experimentos las métricas **han mejorado**, con especial énfasis en el primero y el segundo, que el incremento en mejoría ha sido notable tanto en MSE como en MAE.

- **Experimento 2**: Mejor resultado general (29.62 % MSE, 4.73 % MAE)
- **Persistencia y MAE**: Complica conseguir un MAE mejor - al ser serie de baja variabilidad

3.4.4.15. Jugando con el horizonte

He incluido las métricas R^2 y RMSE porque juntas me ayudan a interpretar mejor los resultados. R^2 me dice qué fracción de la varianza está explicando el modelo - básicamente, cuán bien captura los patrones de los datos. RMSE, por otro lado, me da el error medio en las unidades originales y castiga más los errores grandes, lo que me viene bien para saber la magnitud real de los fallos. Aquí me interesa explorar cómo afecta el horizonte, así que he escogido un conjunto de hiperparámetros que me han dado en todas las pruebas un buen resultado de coste/rendimiento.

Tabla 3.7: Transformer según **horizonte**

Horizonte	MSE	RMSE	MAE	R^2
1	0.0009	0.0300	0.0144	0.9412
3	0.0016	0.0394	0.0210	0.8980
6	0.0022	0.0469	0.0270	0.8491
10	0.0030	0.0549	0.0331	0.7894
15	0.0036	0.0603	0.0375	0.7485

Tabla 3.8: Transformer según **longitud de secuencia**

Longitud	MSE	RMSE	MAE	R^2
5	0.0044	0.0661	0.0432	0.7066
15	0.0030	0.0549	0.0331	0.7908
20	0.0029	0.0536	0.0329	0.7943
30	0.0030	0.0544	0.0344	0.7774
50	0.0034	0.0587	0.0354	0.7782

`hidden_size=128` · `layers=2` · `dropout=0.4` · `batch_size=32` · `lr=0.001` · `epochs=250` · `weight_decay=0.001` · `validation_split=0.3`

El transformer conserva gran parte de la varianza explicada en horizontes cortos, pero cada paso extra introduce incertidumbre acumulada: el R^2 baja de forma suave porque las dependencias se diluyen y el modelo acaba extrapolando con menos información. Con el RMSE podemos ver cómo crece la magnitud del error. Con la longitud de la secuencia, pasa algo parecido: hay una franja óptima ($\approx$ 15–20 pasos) que aporta contexto suficiente sin dispersar la atención; **si miramos menos, falta historia, y si miramos más, el ruido infla el error**.

3.4.4.16. ...y de vuelta a LSTM. Pero ahora: multivariante, multihorizonte y con auto-atención.

De todo lo anterior; varias ideas me parecieron muy relevantes. Por un lado, intentar predecir el siguiente estado únicamente, en este contexto no es buena idea, necesitamos más información y tendencias. **Esto se corrije si agregamos un multihorizonte**. Por otro lado, la auto-atención hace que cada paso o estado sepa donde fijarse dentro de la propia secuencia - **aprenden cómo les influyen otros estados**. Por último, es muy útil el tener una **referencia a superar**, para comparar el modelo debidamente.

En internet hay de todo, para todo; pero he visto que las **arquitecturas predominantes para series temporales** han sido durante mucho tiempo las **LSTM**. Entiendo que mis datos son producidos por mi simulación y quizá en algún punto generan inconsistencias y, entiendo que mi dataset no es enorme. Por ello, sabía que algo tenía que poder hacer para poder tener una LSTM de calidad ya que teóricamente se ajusta más a mi problema con un Transformer - así que intenté aplicar lo aprendido en el proceso y hacer una **LSTM multivariante multihorizonte con atención**. Veamos su arquitectura:

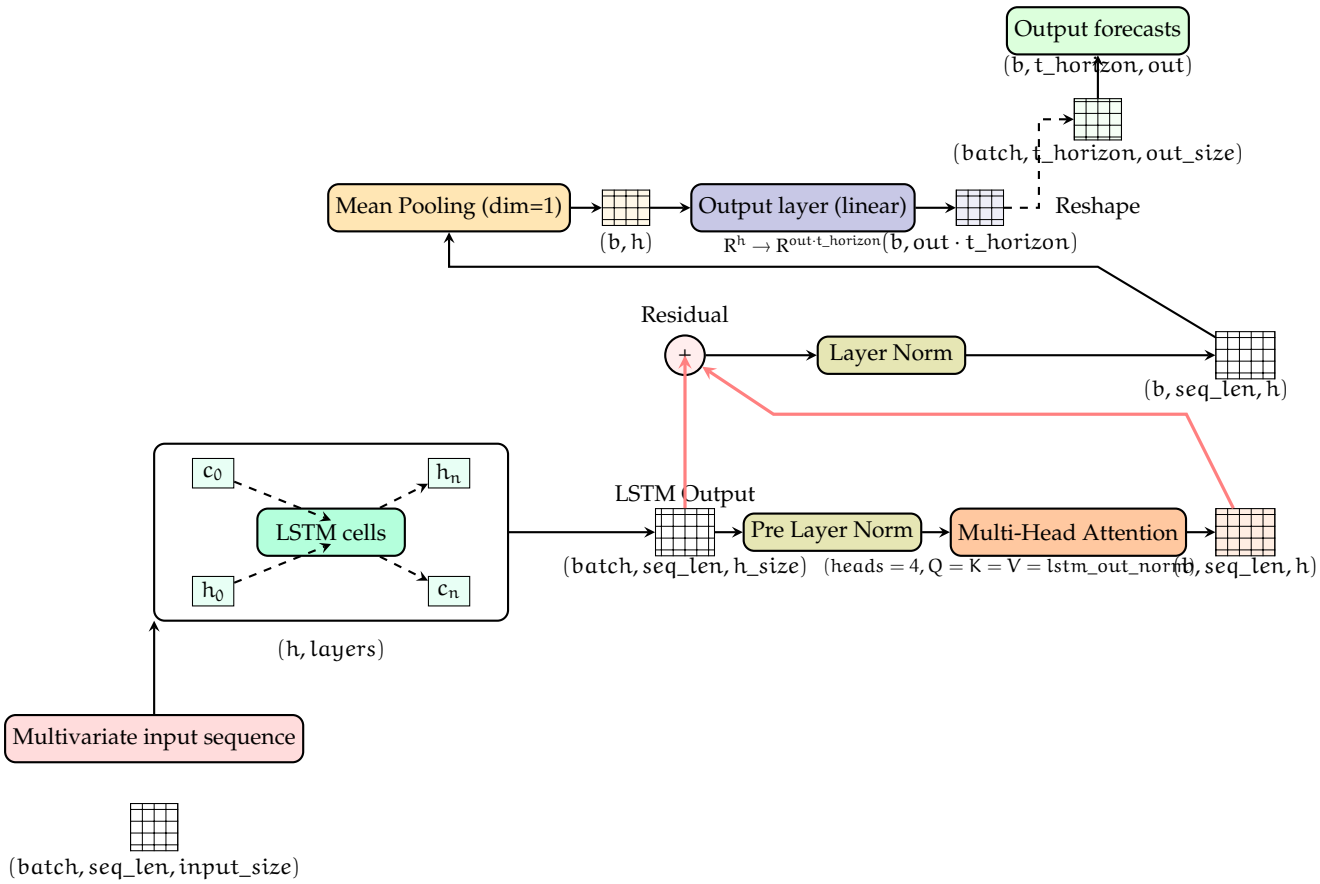


Figura 3.61: Arquitectura de LSTM multivariante mutihorizonte con auto-atención

Arquitectura LSTM multivariante multihorizonte con atención

Codificación secuencial → atención + conex. residual → agregación temporal → proyección multihorizonte

Codificación Secuencial:

- **Núcleo LSTM multicapa:** Este es el núcleo del modelo; con ello capturo relaciones temporales complejas y de largo alcance en los datos. LSTM es conocida por su mitigación del problema de vanishing gradients, con lo que es una elección directa cuando, en un caso como el mio, se quiere capturar tendencias y patrones a largo plazo, como fluctuaciones climáticas o poblacionales (GRU también - y más si el dataset es pequeño - pero como expliqué anteriormente; obtuve resultados muy parecidos).
- **Inicialización de pesos:** Los pesos los he inicializado con algo *standard*: Xavier para las conexiones input-hidden (varianza estable) y Ortogonal para hidden-hidden (preserva norma de gradientes), esto siempre ayuda a mejorar un poco la convergencia.

Atención + residual:

- **Normalización por capa inicial:** La salida de la LSTM, se pasa por una normalización por capa o **layernorm** (para estabilizar el entrenamiento) y se lleva como input a un módulo de atención.
- **Atención multicabezal:** Éste módulo procesa toda la secuencia - la salida se proyecta a los 3 tensores de atención (**key, query y value**) y cada cabezal aprende diferentes patrones temporales. En un enfoque puramente recurrente la información se va comprimiendo en el estado oculto - en un solo vector puede no ser suficiente (apuntes PLN) - sin embargo, con la atención, accedemos a información de cualquier punto de la secuencia ya codificada.
- **Conexión residual:** También dispongo de una **conexión residual** entre la salida de la LSTM y la capa de atención. Es similar a las que se usan en **ResNets** (redes precursoras de las conexiones residuales) y cumple dos funciones: **1)** ayuda con la propagación de gradientes durante el entrenamiento (una vez más, **desvanecimiento de gradientes**) y **2)** combina la **información secuencial** de la LSTM con la **información contextualizada de la atención**. Conceptualmente: output $=$ LayerNorm(lstm_out $+$ attention_out)
- **Normalización por capa final:** La salida combinada, la pasamos una vez más por *otra* **layernorm** para que no se vuelva numéricamente inestable.

Agregación temporal:

- **Pooling temporal:** Soy consciente de que en general, se suele tomar el último estado oculto, ya que comprime información de la secuencia y además hace pasivamente lo que intenté hacer en el modelo transformer - dar más importancia a los últimos pasos de la secuencia. Sin embargo, he encontrado **pequeñas pistas** - principalmente en contexto NLP - que dicen que tanto el **mean como el max pooling suelen dar mejores resultados** (*Maini et al., 2020*[70]), (*Zhou et al., 2016*[71]). Aunque fuera en otro contexto; suficiente para mí para querer explorar y probar: dispongo de un **pooling temporal** sobre toda la secuencia procesada, lo que calcula **la media de los vectores de representación a lo largo del tiempo**. Mi intuición es que el estado oculto puede contener mucho ruido, y quizá el mean pooling lo suaviza.

Proyección multihorizonte:

- **Proyección lineal:** He visto que está muy extendido el uso de modelos autoregresivos, pero ya que había probado el multihorizonte en el transformer, prefería ir a lo seguro y seguir explorando esta idea - además, tiene el gran plus de que **no acumula error**. Con lo que, tras obtener la representación agregada con mean pooling, la proyecto o transformo al **espacio vectorial del horizonte** con una capa lineal (después, los targets serán los `t_horizon` siguientes pasos a la secuencia). Otra ventaja extra - y muy atractiva - es el hecho de que permite que el modelo aprenda **patrones específicos para cada distancia de predicción** (paso en horizonte) - de manera separada (*conceptualmente parecido* a **step embedding** en el modelo transformer). Esto es gracias a distintos pesos de salida para cada paso de predicción.

Modelos relacionados. No he conseguido encontrar otros trabajos que explícitamente utilicen estos mecanismos, he encontrado varios que comparten nociones, como el *Temporal Fusion Transformer* (TFT) (*Lim et al., 2021*[72]). No obstante, al ser un proyecto de exploración y querer *jugar* un poco; opté por intentar diseñar una arquitectura *juntando* los elementos que había visto eran de utilidad - **soy consciente de que, hoy en día, la mejor opción es la de utilizar o fine tunear modelos bien testeados y probados por otros**.

3.4.4.17. Atención: LSTM vs baselines

Creo es interesante analizar los hiperparámetros y hacer diferentes pruebas, pero considero más relevante al contexto el **exponer un enfoque de evaluación sobre decisiones de arquitectura**, como atención, longitud de secuencia, horizonte etc. De las pruebas que he hecho, el conjunto de hiperparámetros base (`h, dropout, lr, wd`) que mejor me ha funcionado de *de manera general* es el que hace de base para el resto de experimentos:

Hiperparámetros Base
`h=128, l=2, dropout=0.4, lr=0.001, wd=0.001`

Tabla 3.9: Attention heads

Heads	Métr.	Mod.	Base	%↑
1	MSE	0.0027	0.0039	29.83
	RMSE	0.0518	0.0622	16.23
	MAE	0.0323	0.0329	1.96
	R²	0.7937	0.7012	13.21
2	MSE	0.0026	0.0039	32.33
	RMSE	0.0509	0.0622	17.74
	MAE	0.0310	0.0335	5.81
	R²	0.8034	0.7012	14.59
4	MSE	0.0025	0.0039	36.51
	RMSE	0.0495	0.0622	20.32
	MAE	0.0306	0.0335	8.65
	R²	0.8180	0.7012	16.66
8	MSE	0.0025	0.0039	34.82
	RMSE	0.0502	0.0622	19.27
	MAE	0.0313	0.0335	6.36
	R²	0.8116	0.7012	15.76
16	MSE	0.0024	0.0039	37.12
	RMSE	0.0493	0.0622	20.70
	MAE	0.0309	0.0335	7.68
	R²	0.8176	0.7012	16.60
32	MSE	0.0025	0.0039	34.43
	RMSE	0.0503	0.0622	19.03
	MAE	0.0317	0.0335	5.28
	R²	0.8104	0.7012	15.57
64	MSE	0.0026	0.0039	33.69
	RMSE	0.0506	0.0622	18.57
	MAE	0.0318	0.0335	5.06
	R²	0.8103	0.7012	15.56

Tabla 3.10: Mejora por horizon

Hori.	Métr.	Mod.	Base	%↑
1	MSE	0.0008	0.0008	5.33
	RMSE	0.0282	0.0289	2.70
	MAE	0.0135	0.0103	-30.56
	R²	0.9460	0.9418	0.45
3	MSE	0.0013	0.0016	20.21
	RMSE	0.0360	0.0403	10.68
	MAE	0.0192	0.0175	-9.89
	R²	0.9064	0.8819	2.78
6	MSE	0.0019	0.0027	29.35
	RMSE	0.0436	0.0518	15.95
	MAE	0.0254	0.0255	0.39
	R²	0.8578	0.7960	7.76
10	MSE	0.0025	0.0039	36.51
	RMSE	0.0495	0.0622	20.32
	MAE	0.0306	0.0335	8.65
	R²	0.8180	0.7012	16.66
15	MSE	0.0032	0.0049	35.39
	RMSE	0.0565	0.0702	19.62
	MAE	0.0364	0.0397	8.16
	R²	0.7645	0.6179	23.73
20	MSE	0.0035	0.0056	38.11
	RMSE	0.0591	0.0751	21.33
	MAE	0.0383	0.0435	12.03
	R²	0.7412	0.5651	31.16
30	MSE	0.0042	0.0066	35.66
	RMSE	0.0650	0.0810	19.79
	MAE	0.0427	0.0485	11.99
	R²	0.6997	0.5057	38.36

Tabla 3.11: Mejora por longitud

Long.	Métr.	Mod.	Base	%↑
5	MSE	0.0030	0.0040	24.65
	RMSE	0.0546	0.0629	13.20
	MAE	0.0340	0.0335	-1.28
	R²	0.7928	0.7035	12.69
15	MSE	0.0025	0.0039	36.51
	RMSE	0.0495	0.0622	20.32
	MAE	0.0306	0.0335	8.65
	R²	0.8180	0.7012	16.66
20	MSE	0.0025	0.0038	33.67
	RMSE	0.0504	0.0618	18.55
	MAE	0.0311	0.0329	5.60
	R²	0.8093	0.7011	15.43
30	MSE	0.0027	0.0040	32.34
	RMSE	0.0519	0.0630	17.74
	MAE	0.0319	0.0333	4.21
	R²	0.8000	0.7024	13.90
50	MSE	0.0034	0.0042	18.94
	RMSE	0.0583	0.0648	9.97
	MAE	0.0371	0.0345	-7.45
	R²	0.7727	0.7063	9.40

- **Nota**: He hecho pruebas a incrementar el tamaño de los vectores de represntación conforme subía el valor del número de cabezales de atención, pero no he notado ninguna diferencia relevante, así que no documento.
- **Nota2**: Los baselines de las tablas 3.10 y 3.11 cambian, al afectar durante la construcción del modelo estático.

Tabla 3.12: Comparación LSTM vs Baseline: predicción multi-horizonte (10 pasos)

Step	Target	LSTM pred	Baseline pred	LSTM MAE	Baseline MAE	LSTM MSE	Baseline MSE
1	[4.00, 10.00, 73.00, 0.75, 0.40]	[2.82, 10.07, 69.38, 0.75, 0.54]	[2.00, 10.00, 73.00, 0.73, 0.39]	1.00	0.41	2.90	0.80
2	[4.00, 10.00, 72.00, 0.75, 0.43]	[2.86, 10.09, 69.29, 0.75, 0.55]	[2.00, 10.00, 73.00, 0.73, 0.39]	0.81	0.61	1.74	1.00
3	[3.00, 10.00, 71.00, 0.75, 0.46]	[3.01, 10.29, 68.74, 0.75, 0.71]	[2.00, 10.00, 73.00, 0.73, 0.39]	0.56	0.62	1.05	1.00
4	[2.00, 10.00, 69.00, 0.74, 0.50]	[2.99, 10.42, 68.33, 0.75, 0.65]	[2.00, 10.00, 73.00, 0.73, 0.39]	0.45	0.82	0.33	3.20
5	[2.00, 12.00, 67.00, 0.75, 0.51]	[3.17, 10.63, 67.61, 0.76, 0.85]	[2.00, 10.00, 73.00, 0.73, 0.39]	0.70	1.63	0.74	8.00
6	[2.00, 12.00, 67.00, 0.75, 0.54]	[3.19, 10.87, 67.27, 0.76, 0.73]	[2.00, 10.00, 73.00, 0.73, 0.39]	0.56	1.63	0.56	8.00
7	[2.00, 12.00, 67.00, 0.75, 0.56]	[3.20, 11.06, 67.76, 0.76, 0.92]	[2.00, 10.00, 73.00, 0.73, 0.39]	0.66	1.64	0.61	8.01
8	[2.00, 12.00, 67.00, 0.75, 0.58]	[3.49, 11.18, 68.01, 0.77, 1.07]	[2.00, 10.00, 73.00, 0.73, 0.39]	0.77	1.64	0.83	8.01
9	[2.00, 12.00, 68.00, 0.75, 0.58]	[3.53, 11.24, 67.58, 0.77, 1.23]	[2.00, 10.00, 73.00, 0.73, 0.39]	0.68	1.44	0.70	5.81
10	[2.00, 12.00, 67.00, 0.76, 0.57]	[3.33, 11.38, 67.40, 0.77, 1.64]	[2.00, 10.00, 73.00, 0.73, 0.39]	0.68	1.64	0.69	8.01
Promedio total				**0.687**	**1.208**	**1.016**	**5.184**
Mejora LSTM vs Baseline					**43.1 %**		**80.4 %**

Epoch 250. Mejor modelo LSTM vs Baseline.
Target: `[prey_population, predator_population, flora_count, ecosystem_stability, avg_stress]`.
MAE y MSE calculados sobre **todas** las variables de salida.

Si indagamos en los cálculos durante el proceso de entrenamiento (3.12), en ésta última época podemos ver que el LSTM captura tendencias que el baseline no. Por ejemplo, cuando la población de depredadores salta de 10 a 12 entre los pasos 4-5, el baseline, por su definición de persistencia, mantiene el valor constante, pero el **LSTM anticipa gradualmente la transición**. No obstante, creo que una barrera que tengo es mi dataset, la calidad de los datos quizá no es del todo buena y adeas, necesito muchos más para que el modelo pueda aprender mejor (≈35,000 muestras actualmente).

Con todo esto, podemos observar que los mejores hiperparámetros son los que consiguen un equilibrio entre *ver lo suficiente* y *no perderse con el ruido*. La atención tiene un impacto positivo, con *attention heads* (4-16) consigo que el modelo mire la serie desde varios ángulos - añadir más cabezas apenas aporta señal extra y, en cambio, complica la optimización. Respecto al horizonte, el mejor es el de 20, pero para las pruebas he utilizado (≈10 pasos); funciona muy bien, captura la dinámica; no es tan corto que evita que se pierda contexto, ni tan largo que termina especulando y con mucha incertidumbre. Por último, la longitud de secuencia ha permanecido en 15 y, ocurre lo mismo - *ni mucho ni poco*: ventanas cortas ignoran patrones latentes y ventanas muy largas mezclan información de segmentos, y esto dispersa la atención del modelo.

Mejor modelo:

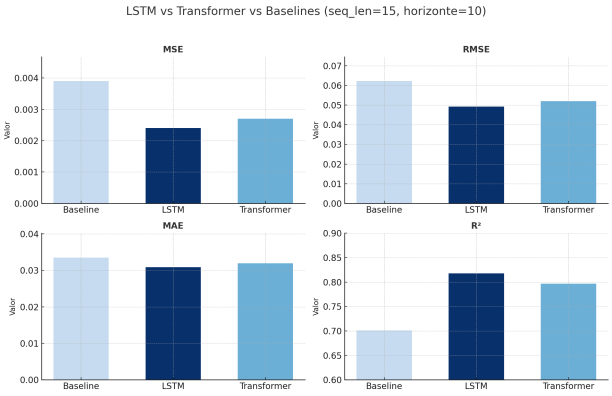


Figura 3.62: Comparativa mejores modelos

3.4.5. Módulo simbólico: de reglas a grafos

He diseñado los módulos simbólicos tal que tengan una interfaz común, para que puedan ser intercambiables e inyectables a tiempo real. Les paso la información tanto a nivel de bioma como de especies a través de un servicio de datos especializado (que, como todos, se nutre de los recolectores). Como se mostró en la figura 3.4.3, por un lado el módulo neural invocará el método `predict` y nos dará sus predicciones basándose en las secuencias históricas de la simulación, y por otro el módulo simbólico hará inferencia para darnos feedback de su análisis sobre el estado actual del bioma, especies...etc. - lo traduce en **recomendaciones de acción**.

Ya con estas dos masas de información, ahora las mezclamos y producimos recomendaciones de acción ponderando los feedbacks. El componente de integración, no es tan avanzado como los standard en IA Neurosimbólica - pero funciona bastante bien para combinar el *feedback* ambos

módulos y decidir cómo intervenir para intentar estabilizarlo. Los datos que reciben ambos módulos son proporcionados por los recolectores a tiempo real durante la simulación - están en memoria.

3.4.5.1. Base de conocimiento: sistema basado en reglas

Para este primer enfoque, he implementado la vertiente más sencilla posible; un sistema de reglas que analiza métricas del bioma y da recomendaciones basadas en umbrales que he ido definiendo manualmente. Funciona bastante bien para lo básico que es. Está estructurado en cuatro categorías: **estado de especies, dinámicas depredador-presa, estabilidad del ecosistema, y condiciones climáticas**. Las reglas son bastante intuitivas, por ejemplo:

- Si veo que la población de una especie se desploma por debajo de cierto número, la marco **en peligro**
- Cuando el estrés medio se dispara, pongo a la especie en estado **estresada**
- Si hay demasiados depredadores vs pocas presas, **se detecta un desequilibrio trófico**
- Cuando la biodiversidad baja mucho, **recomiendo introducir más especies**.

A modo de ejemplo rápido; para saber si una especie está a punto de extinguirse uso algo tan simple como:

$$\text{población} < \theta_{\text{crítico}} \Rightarrow \text{especie\_en\_peligro} \tag{3.28}$$

El umbral $\theta_{\text{crítico}}$, puede ser un determinado porcentaje de la cantidad con la que la especie comenzó la simulación, o también un número constante con valor 7. Tras muchas pruebas, he observado que cuando una especie baja de ese número - su destino ha sido escrito, tiende a extinguirse.

También he implementado una regla para detectar cuando hay mucha presión por parte de depredadores:

$$\frac{\text{población\_depredadores}}{\text{población\_presas}} > 0{,}3 \Rightarrow \text{presión\_depredación\_alta} \tag{3.29}$$

A parte de identificar el problema, genera recomendaciones con niveles de prioridad. Lo bueno de

este enfoque es que es transparente - es muy fácil hacer cambios y detectar qué va mal. El problema es que **no captura interdependencias complejas** del bioma, y además, las especies no funcionan de forma aislada, **hay toda una red trófica que este sistema no puede entender bien**. Se pueden ver el resto de reglas implementadas en **Reglas del módulo simbólico** (Apéndice, p. 125).

3.4.5.2. Sistema basado en grafos

En este segundo enfoque **modelo el ecosistema como un grafo dirigido** - aplico técnicas de análisis de redes para intentar extraer información sobre la estructura del bioma. Desde primer curso - esto es algo que he apreciado mucho, el aprender teoría de grafos desde el primer semestre me hizo ver las posibilidades que tenían para modelado de problemas, y me hizo un pequeño *click*, aunque lo que viéramos fueran los fundamentos. Desde entonces, incluso en el trabajo he llegado a modelar varios problemas con grafos, lo cual antes de pasar por la universidad no hubiera podido pensar fuera de mi *FOV de herramientas*.

Bien, con esto, ahora puedo identificar propiedades que no detectaría el enfoque anterior. En este grafo:

Estructura del grafo

- **Nodos** - especies y factores ambientales (temperatura, humedad, etc.)
- **Aristas** - interacciones (depredación, consumo de recursos, impacto ambiental)
- **Peso de las aristas** - intensidad de la interacción (ahora mismo *hardcodeadas*)

Con este enfoque puedo calcular **métricas de centralidad**, con las que puedo identificar especies clave (*keystone*) en el bioma, por ejemplo; para cada especie se calculan métricas como *degree centrality* (conexiones directas), *betweenness centrality* (especies que actúan como puente), y *eigenvector centrality* (importancia global en la red) - con esto conseguimos detectar esas *keystone*. Con esto, también podemos identificar niveles tróficos (si calculamos la distancia de cada especie a productores primarios).

Se pueden ver todos los algoritmos utilizados aquí en la tabla **Apéndice 4.3: Métricas devueltas por GraphBasedSymbolicModule**. Estos algoritmos no los he programado yo, he utilizado la librería de **networkx**, que expone una interfaz para poder construir grafos y utilizar un gran abanico de algoritmos.

3.4.5.3. Integración ponderada de los módulos: Late fusion (asimétrico)

A la hora de combinar resultados de ambos módulos, me ha parecido mejor opción el utilizar **Late fusion** - cada módulo, neural y simbólico, trabaja por su cuenta y luego junto sus resultados. Entonces, el proceso mezcla: predicciones numéricas del módulo neural, orientadas a poblaciones y parámetros del bioma, con **recomendaciones** de acción del módulo simbólico. La idea en la que me basé es en intentar buscar un **nexo entre ambos módulos mediante comprobaciones** y, con esto tomo la decisión. Para ello los integro con un sistema de pesos donde doy más importancia a uno u otro según el caso - por defecto uso 40 % neural y 60 % simbólico, aunque esto lo puedo ajustar en tiempo real e incluso se podría hacer adaptativo por variables.

$$R_{\text{integrado}} = \alpha \cdot R_{\text{neuronal}} + (1 - \alpha) \cdot R_{\text{simbólico}} \tag{3.30}$$

Claro, lo interesante de IA Neurosimbólica es que el resultado final funciona mejor que si tomamos ambos enfoques por separado.

3.4.5.4. Sistema de intervenciones

El resultado final de todo el proceso visto, es un conjunto de intervenciones - con ellas se equilibra el bioma. He desarrollado dos tipos principales de intervenciones:

Ajuste de ciclos evolutivos: En comunicación con el agente evolutivo de cada especie, se altera el tiempo o ciclo al que evolucionan; el ajuste lo calculo en la fase de integración y depende de diferentes factores; desde tendencias poblacionales, hasta demasiada presión por parte de los depredadores.

Nuevas entidades: Inyecta nuevas entidades en el bioma cuando es necesario, pero con control para no sesgar la evolución y **preservar la diversidad genética**; hago una `clonación de genes` de individuos de última generación, para que se puedan seguir propagando. También en el proceso de integración es cuando se calculan los factores, como el número de nacimientos, etc.

Intervención

Veamos un ejemplo breve: en un **desequilibrio depredador-presa** con dominancia de depredadores y niveles críticos de presas, el agente ejecuta varias acciones **simultáneas** (no siempre ejecutará todas, depende) :

- Ordena que se **acelere la evolución** de presas para adaptarse a la presión depredadora.
- A una especie depredadora, le **ralentiza la evolución**.
- **Nuevas presas/clones** inyectadas en población.

Se puede consultar la tabla de las intervenciones posibles aquí **Sistema de intervenciones neurosimbólicas** (Apéndice, p. 127).

A veces se pueden ver cosas interesantes, pero el factor que más complicado me ha parecido ha sido el *timing* - si intervengo demasiado pronto, el sistema natural no tiene oportunidad de autorregularse; si intervengo demasiado tarde, pérdida irreversible de especies. No obstante, hay que tener en cuenta que el modelo actual de Echoes of Gaia, como he mencionado en otras ocasiones, toma una vertiente simplista para representar ciertas áreas de la naturaleza. Especifico al final posibles lineas de mejora.

Guardián de Gaia

Veamos unos resultados para comprobar cómo nuestro agente **balancea y protege los ecosistemas**. Se han lanzado simulaciones con y sin el agente de Equilibrium - el campo evento de finalización indica si la simulación llegó a lo programado o no. Si el bioma ha sufrido colapso, como extinción de fauna etc., puede terminar antes.

Los valores que se muestran, son los valores **al terminar** la simulación, ya sea de manera prematura por colapso, o bien que se ha llegado al tiempo de simulación configurado.

Tabla 3.13: Comparativa de rendimiento del sistema neurosimbólico en simulaciones

Sim ID	Agente Equilibrium	Estrategia	Eventos tot.	Evento finalización	Puntuación bioma	Índice biodiv.	Flora/ fauna	Presa/ pred.	Estrés medio
1	✓	`Grafos`	7200	7200	6.50	0.734	31/14	4/10	6.48
2	✓	`Reglas`	7200	7200	6.32	0.683	86/17	7/10	0.99
3	×	-	7200	1620	5.72	0.936	280/0	0/0	1.67
4	×	-	7200	7200	5.11	0.813	327/2	2/0	0.98
5	✓	**`Grafos`**	**5000**	**5000**	**8.60**	**0.958**	**56/51**	**34/17**	**5.15**
6	✓	`Grafos`	1000	1000	7.05	0.953	56/52	14/38	4.93
7	×	-	1000	1000	5.41	0.837	252/4	4/0	4.48
8	×	-	5000	5000	5.28	0.822	249/3	3/0	0.68

Sim ID	Pop. balance	Biodiversity	Ecosystem health	Observaciones
1	0.822	0.467	0.630	Equilibrio moderado
2	0.530	0.616	0.695	Eequilibrio moderado
3	0.200	0.668	0.739	Sin agente: extinción fauna
4	0.212	0.456	0.715	Sin agente: colapso predadores
5	**0.872**	**0.939**	**0.911**	**Óptimo**
6	0.864	0.576	0.689	Buena estabilidad
7	0.231	0.618	0.632	Sin agente: desequilibrio severo
8	0.224	0.611	0.722	Sin agente: desequilibrio severo

3.4.6. Intervenciones de mejor simulación (Sim 5) - biome type: Taiga

T. Sim.	Tipo	Especie	Factor	Justificación
29.68	Generate	`winterberry`	$N = 15$	Flora-herbívoro desbalanceado (ratio: 0.2)
29.68	Evolution	`lynx`	$\alpha = 0{,}985$	Control poblacional de presas
29.68	Evolution	`moose`	$\alpha = 1{,}015$	Predicción neural. Decisión híbrida: ↑ ratio predador/presa
29.68	Keystone protection	`spruce_tree`	$\alpha = 0{,}990$	Especie clave (centralidad de red)
68.55	Generate	`spruce_tree`	$N = 10$	Flora-herbívoro desbalanceado (ratio: 0.32)
68.55	Evolution	`lynx`	$\alpha = 0{,}990$	Control poblacional de presas
180.87	Generate	`winterberry`	$N = 10$	Flora-herbívoro desbalanceado (ratio: 0.45)
180.87	Evolution	`wolf`	$\alpha = 0{,}995$	Optimización predatoria
180.87	Keystone protection	`spruce_tree`	$\alpha = 0{,}990$	Especie clave (centralidad de red)

Tabla 3.14: Resumen (muestra parcial). Factores $\alpha < 1$ aceleran la evolución, $\alpha > 1$ la ralentizan.

Especie	Generate	Evol.	Key prot.
`winterberry`	4	1	2
`spruce_tree`	2	0	6
`moose`	0	9	0
`lynx`	0	8	0
`arctic_moss`	1	0	0
`wolf`	3	2	0

Tabla 3.15: Matriz de frecuencias, de toda la simulación

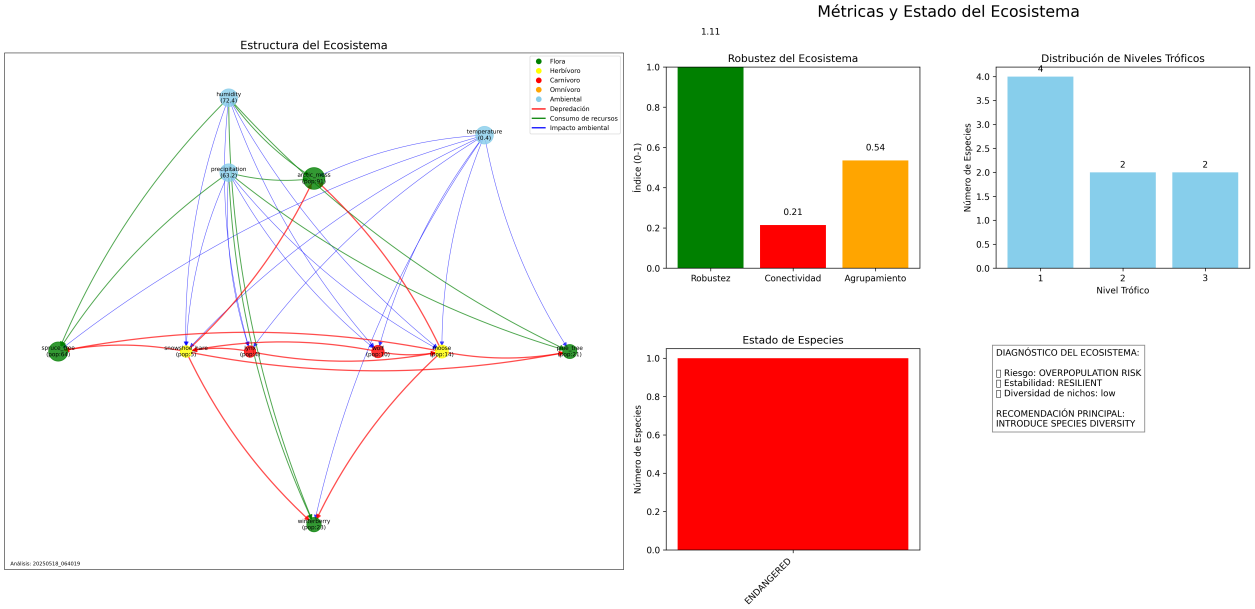
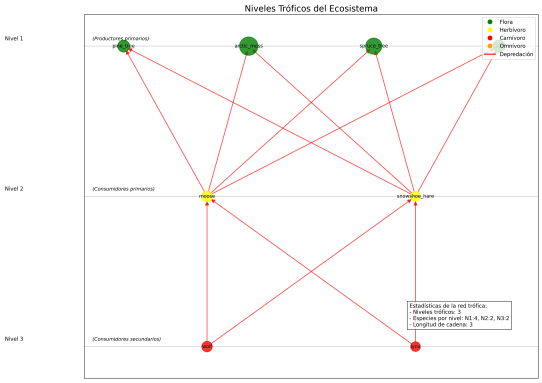


Figura 3.64: Grafo y métricas del bioma

Este es el grafo de temperatura, humedad y precipitación. Las aristas azules modelan la presión de estas variables sobre cada especie, las verdes el consumo vegetal y las rojas la predación. Pensé en esta forma para intentar representar el estado del bioma con un grafo, y efectivamente, en estos escenarios vemos cómo los factores abióticos influyen en la dinámica sin tocar directamente la estructura biológica.

Algo a destacar: alta robustez (1.0) pero baja conectividad (0.21). Esto me dice que hay pocas especies con roles muy específicos que, aunque son resilientes por sí solas, crean un sistema simple. Para medir la robustez elimino especies aleatoriamente - aquí, perder el 20 % de las especies apenas toca la estructura, lo que implica que las especies restantes mantienen las funciones ecosistémicas básicas.
Baja conectividad = pocas interacciones; de todas las posibles solo ocurre el 21 %. En un ecosistema de 8 especies como éste, es relativamente poco; podemos inferir que las especies viven bastante aisladas, con pocas dependencias entre ellas.



Efectivamente, la estructura trófica nos confirma la simplicidad: cuatro especies productoras en el nivel 1, dos herbívoras en el nivel 2 y dos carnívoras en el nivel 3; faltan niveles intermedios y especialistas.

Figura 3.65: Estructura trófica del ecosistema

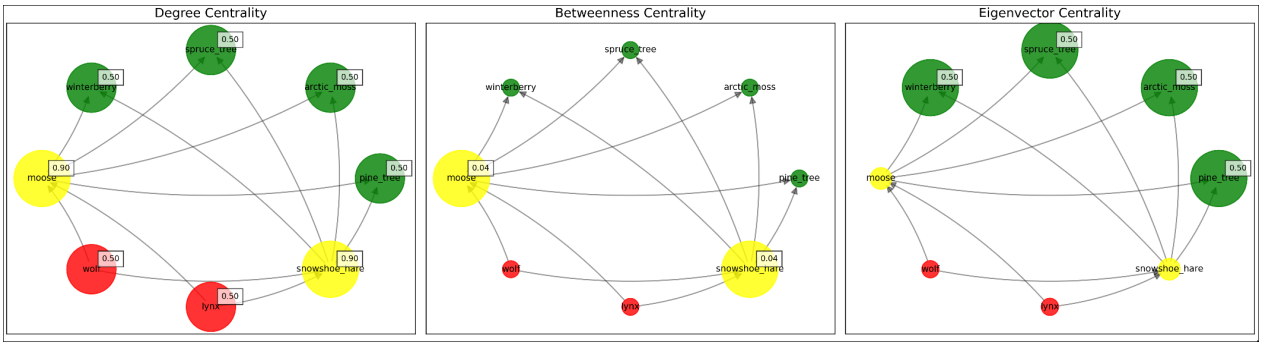


Figura 3.66: Centralidad de los nodos en la red trófica

Degree centrality

En el análisis de centralidad, *moose* y *snowshoe_hare* dominan en degree centrality (0.90) como **generalistas** que consumen múltiples especies vegetales. La flora es hacia donde se converge en los niveles tróficos.

Betweenness centrality

Efectivamente, en betweenness centrality *moose* y *snowshoe_hare* actúan como los únicos puentes entre depredadores y productores primarios. También podemos ver un riesgo de sobrepoblación en las métricas y esto surge porque hay pocos depredadores para controlar herbívoros (3.64. Los herbívoros pueden crecer exponencialmente hasta **terminar toda la vegetación** (esto a largo plazo **afecta al clima** también).

Eigenvector centrality

Viendo el **eigenvector centrality**, toda flora tiene 0.50 - ninguna domina pero, tienen mucha importancia ya que en este contexto lo que éste algoritmo mide es: *si te come o te come alguien muy conectado, tu propia influencia en la red aumenta*. Con esto se puede ver que la flora sostiene a herbívoros, y a su vez sirven de presa para depredadores: si la flora desaparece - **provoca cascada trófica**. Ahora mismo las relaciones son simples, pero se puede ver la enorme utilidad de esto si se granuralizan más en un futuro, que es para lo que está pensado el framework.

El sistema nos ha recomendado introducir diversidad de especies. Claro, esto idealmente resuelve múltiples problemas: más carnívoros controlarían herbívoros, más herbívoros especializados reducirían competencia, y más productores diversifican la base alimentaria. **El resultado sería mayor conectividad y se mantiene la robustez actual (conexiones balanceadas), lo cual, mejoraría ese 0.21 de conectividad que vimos al comienzo**.

CAPÍTULO 4

Conclusiones

4.1. Conclusiones

A modo de recapitulación, considero que el principal logro de Echoes of Gaia es haberse podido llevar a cabo - con sus más y sus menos, pero he conseguido diseñar y construir un *framework* holístico que funcione como laboratorio ecológico virtual. El proceso no ha sido trivial - en muchas ocasiones las piezas no han sido fáciles de encajar y un mínimo cambio producía una cascada de errores que había que resolver.

He disfrutado mucho de realizar el modelado computacional de los procesos biológicos y las métricas ecológicas, pero a su vez me ha hecho ver que parece no tener fin. Es decir, la naturaleza es tan compleja que hay que hacer un enorme filtrado y decidir qué se quiere modelar y representar en el bioma. Me ha hecho ver que todo lo que haga resultará simplista comparado con la inmensidad del mundo real, pero creo que he conseguido capturar algunos patrones básicos.

En cuanto a optimización, ha sido importante el reorganizar la arquitectura para poder paralelizar y optimizar cómputo cuando he necesitado cálculos intensos. La vectorización ma ha proporcionado mejoras del 44-49 % en rendimiento - no es algo trivial cuando manejas miles de entidades con componentes complejos.

El uso de Reinforcement Learning para el agente climático ha terminado siendo muy útil y adecuado para lo que necesitaba, aunque en un primer momento puede no parecer la opción más directa. Al no disponer de datasets climáticos, creo que ha sido una muy buena elección para el proyecto y gracias a ello he podido tener un ente inteligente que aprende a planificar y adaptarse, pero con el objetivo fijado en los factores medioambientales del bioma que se esté simulando. Con el agente de fauna he aprendido lo importante que es guiarle correctamente durante el entrenamiento, además de tratarle con cariño y no penalizar en exceso; por ejemplo con el truco que he utilizado de implementar un sistema artificial que activa entidades cercanas con comportamientos presa-depredador para forzar que el agente aprenda dinámicas diferentes - al final termina entendiendo estas relaciones y las ejecuta de manera bastante convincente.

A la hora de elegir arquitecturas de redes neuronales, he podido ver de primera mano por qué nos insistian en asignaturas como AARN y PLN sobre la importancia de estudiar bien el problema y los datos. En mi caso, al haber hecho un viaje experimental de LSTM a transformers, y volver a LSTM, he visto cómo mantener un modelo más simple pero adaptado puede dar mejores resultados. La LSTM con atención que diseñé al final ha hecho que el agente de equilibrium mejore el balance ecosistémico y haga las simulaciones más estables.

Algo que puede parecer obvio, pero que ha terminado siendo de grandísima ayuda ha sido el implementar un modelo baseline de persistencia para tener una referencia clara con la que comparar mis modelos. Sin esto, ver que el MSE baja no me decía mucho, ya que el error es relativo a la escala de los datos.

Es muy complicado encontrar un balance con tanto cálculo de manera que el sistema se autorregule - constantemente hay que estar ajustando parámetros y umbrales para evitar que algo se descontrole.

En general, desde el punto de vista ecológico, las sinergias conseguidas entre estos agentes multi-paradigma, son dinámicas que nos pueden servir para plantear escenarios e hipótesis y observar sus resultados, sin tener que esperar el tiempo que requeriría en la naturaleza. Al fin y al cabo, junto con la facilidad de poder generar estos escenarios, era el objetivo principal del proyecto.

4.2. Limitaciones y líneas de trabajo futuro

Las líneas de trabajo futuro están precisamente basadas en las limitaciones actuales, así que las fusiono para comentar qué me gustaría que fuera mejorado:

- **Modelado más realista:** El sistema actual usa dietas genéricas - un herbívoro puede comer cualquier flora y los depredadores cualquier presa. Me gustaría implementar especializaciones alimentarias y preferencias específicas. También querría desarrollar comportamientos de grupo o incluso modelar la radiación solar explícitamente, que afecte a la fotosíntesis según la nubosidad y elementos similares.
- **Coevolución entre especies:** Ahora mismo cada especie evoluciona por separado. Una idea que me atrae es la de implementar carrera armamentista evolutiva entre depredadores y presas, algo como la *Red Queen Hypothesis*; cada mejora en una especie presiona a la otra a adaptarse.
- **Optimización de rendimiento:** Pese a las mejoras de vectorización, simular múltiples agentes con IA es computacionalmente intenso. Creo que sigue siendo muy lento y debería de poder mejorarse. Lo primero sería hacer un perfilado para detectar bottlenecks y hacer algún cambio en la arquitectura y estructuras de datos para reducir complejidad donde sea necesario. Pero, sería muy buena idea el explorar paralelización real o compilación JIT; podría abrir la puerta a simulaciones mucho más grandes (Numba, es simple y con decoradores `@jit`) - pero sobre todo, llevar el proyecto a un lenguaje compilado como C++.
- **IA Neurosimbólica más avanzada:** Mi implementación actual es básica comparada con lo que se puede hacer. Las GNNs (Graph Neural Networks) podrían analizar la red trófica de manera mucho más rica, y bucles de realimentación más estrechos entre módulos neural y simbólico darían mejores resultados.
- **Curriculum learning:** Me habría gustado implementar para la entrega técnicas como curriculum learning en el contexto de Reinforcement, y hacer que el adaptador provisione de mapas progresivamente más grandes, y que el FOV del agente vaya cambiando también conforme avanza el entrenamiento. Esto lo estabilizaría y puede hacer que el agente aprenda mejor ciertos patrones. También, durante el entrenamiento, escojo un número de generación aleatoria; quizá un truco que podría funcionar bien en ésta línea es el de tomar primero generaciones aleatorias entre 1-3, e ir ampliando la cota alta gradualmente. Al final, todo esto puede guiar al agente a ir aprendiendo poco y según se va preparando - darle cosas más difíciles, en lugar de darle algo demasiado complicado cuando sabemos que aún no está lo suficientemente preparado.

- **Aclimatación térmica gradual:** No sólo adaptación por evolución, si no tener un sistema para que las entidades puedan adaptar su temperatura óptima y resistencias (cuando exista exposición prolongada a ciertas condiciones), podría proporcionar una plasticidad a las especies, y creo que podríamos observar situaciones o respuestas más realistas a cambios.
- **Factor humano:** Me hubiera encantado poder introducir esto, pero desgraciadamente no he tenido tiempo para ello. Introducir un factor que indique la huella que deja el ser humano en un ecosistema (con su extensión, contaminación...etc), sería algo muy interesante y que enriquecería el propósito del proyecto y sus posibles estudios de hipótesis.

Mencionar también, que algo que me ha parecido complicado, ha sido el, tras haber desarrollado todo el proyecto, tener que realizar el ejercicio mental de reorganizar paso a paso todos los elementos, procesos, componentes... y pensar una forma de contarlo de manera ordenada para que alguien que no esté en mi cabeza pueda entenderlo. Cada vez que quería contar algo, existían dependencias no resueltas, estos es, algo que estaba asumiendo se conocía por parte de la lectora o el lector, y resultaba ser que no. He intentado exponerlo lo más ordenado, claro y conciso posible. Disculpad si no lo he conseguido.

4.3. Reflexión personal

Para mí, este proyecto ha sido el summum final de mi paso por la universidad. Ha supuesto un viaje de aprendizaje intenso que he disfrutado muchísimo - me ha permitido aplicar conocimientos de la carrera pero también explorar áreas de la IA que me resultan atractivas e interesantes y tenía en cola.

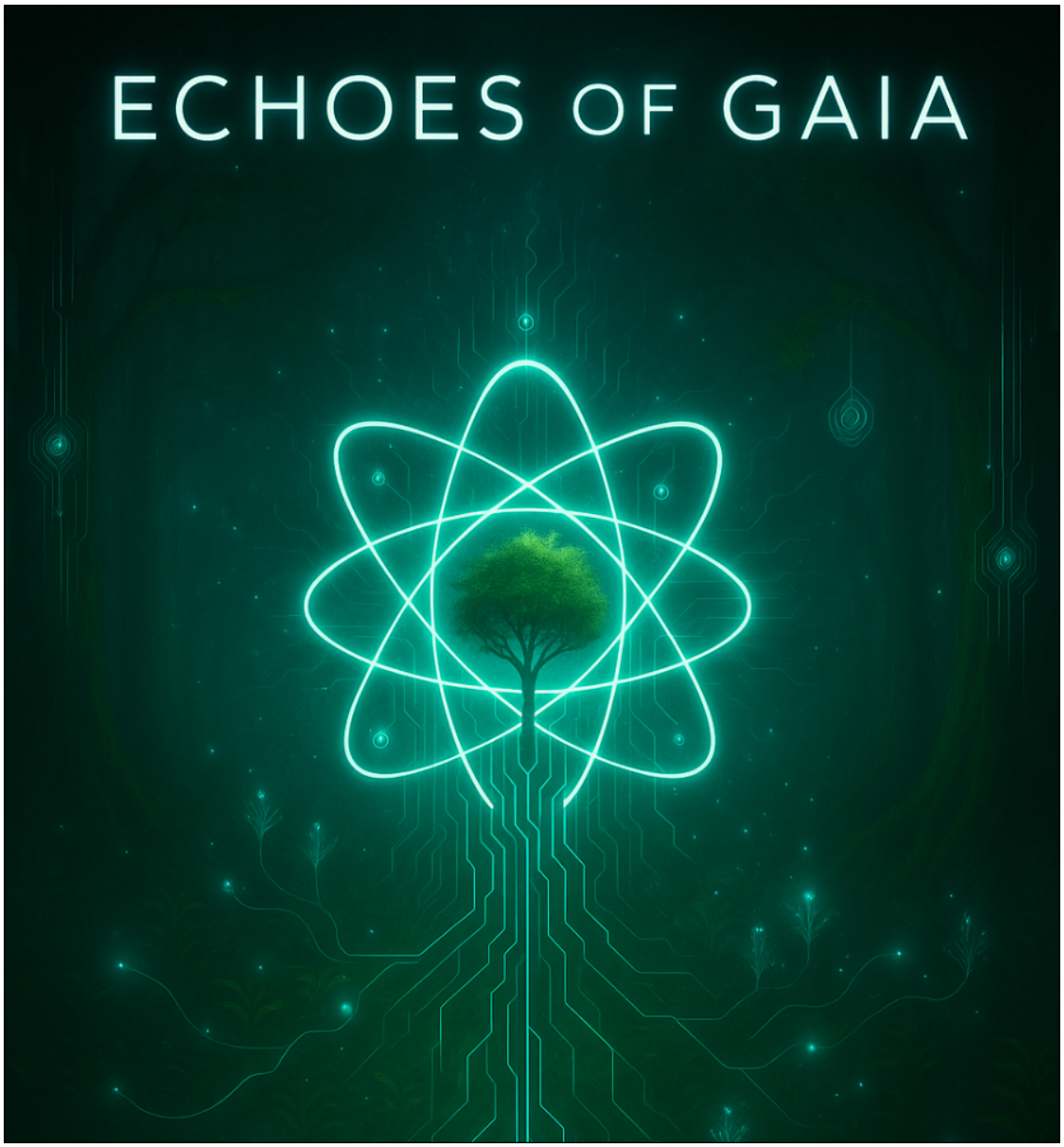
Construir algo desde cero con tantos sistemas interconectados ha sido tedioso ya que, constantemente había que estar ajustando para que el agente evolutivo no se descontrolara, o para que el clima no terminara en valores irrealistas. Por muy pequeño que fuera un cambio en un componente - a veces provocaba cascadas de efectos en todo el bioma. Ya antes de comenzar asumía que balancear un sistema holístico tan grande me daría dolores de cabeza, y ha resultado ser así; pero cuando funciona y ves emerger comportamientos complejos... **es muy gratificante**.

Estoy satisfecho con el resultado - **consciente de sus limitaciones pero también de su potencial** como herramienta de exploración.

La universidad ha cambiado mucho mi manera de pensar. No ha sido fácil a estas alturas de la vida compaginar todo, pero me he volcado muchísimo y ha sido una de las mejores decisiones que he tomado; para mí ha sido un ejercicio duro de autosuperación. Me ha demostrado que nunca es tarde y, durante el desarrollo de este proyecto he comprobado en multitud de ocasiones lo mucho que me ha influido - ahora enfoco los problemas de formas totalmente diferentes, y eso es precisamente lo que buscaba. Herramientas, frameworks, lenguajes de programación van y vienen, pero esa elasticidad y capacidad de pensar de forma más abstracta, de generalizar que te da estudiar la carrera; eso - aunque haya que continuar entrenando - se queda.

Espero que Echoes of Gaia demuestre mi pasión por la computación y la Inteligencia Artificial, sería precioso y me haría mucha ilusión que pueda servir de inspiración o base para futuros desarrollos y/o proyectos.

Muchas gracias por todo.


ECHOES OF GAIA

Apéndice

Metodológia y desarrollo

Durante el desarrollo he intentado seguir una metodología estructurada; primero hice una fase de análisis e investigación para disponer de una noción general y además refrescar o superficilamente investigar sobre algunos conceptos: motor de simulación, algoritmos genéticos, Reinforcement Learning y, sobre todo, procesos ecológicos y biológicos; para ver qué y cómo podría incorporarlo (el sistema neurosimbólico surgió *a tiempo real*, durante el desarrollo). Este primer análisis me fué bastante útil para tener una visión global, y gracias a él diseñar la arquitectura fué más fluido, así como definir cada componente y sus tareas. Planificación del proyecto disponible aquí.

Para ello me centré en diferenciar responsabilidades de cada sistema, dar forma a interfaces/APIs que harían de nexos entre módulos, integrar los patrones de diseño que consideré serían apropiados etc. La idea era crear una base sólida para después desarrollar cada parte de manera independiente.

El proceso que (más o menos) seguí para cada área específica: investigación del tema, diseño de integración con el resto de sistemas, implementación, pruebas exhaustivas y generación de plots/visualizaciones/snapshots para analizar resultados cuando era relevante.
Con esto he podido ir construyendo el framework de manera incremental, para tener validada cada parte antes de continuar - aunque esto es lo teóricamente ideal, nunca/casi nunca ocurre y lógicamente en varias ocasiones he tenido que revisitar sistemas ya implementados que habían sufrido de nuevos bugs. No obstante, tener plots y métricas para cada subsistema, junto con un debugging exhaustivo, ha ayudado a hacer *pin-point* de los problemas.

Al principio empecé implementando un sistema de tests unitarios para intentar asegurarme de que no se rompía nada con actualizaciones, pero al estar solo desarrollando, más la cantidad de trabajo que me esperaba y que además me ralentizaba al ser nuevo con Python - decidí no continuar.
No estimé horas, si no *milestones* o fechas que simulen diferentes sprints. No obstante, he alterado el órden en alguna ocasión (por ejemplo, el visor lo dejé para más tarde). Veamos una tabla orientativa:

Milestone	**Planificado**	**Real**	**Notas**
Simulación y Bioma	1–15 Feb	3–18 Feb	Terminado (+3 d)
Visor de datos básico	16–20 Feb	27 Feb	Retraso 1 sem (por alteración del orden)
Algoritmos genéticos + RL	21 Feb–18 Mar	28 Feb–11 Abr	Retraso 3 sem (por alteración del orden)
Research Hub	19 Mar–2 Abr	11 Feb	Adelantado (-5 sem)
Modelos Deep Learning	3–21 Abr	15–23 Abr	Retraso leve (+2 d)
Demo videojuego	22 Abr–20 May	—	Omitido al final
Memoria y análisis	20 May–22 Jun	23 Abr–1 Junio	Adelantado

Tabla 1: Comparativa de fechas, orientativo

He ido incluyendo algunas funcionalidades nuevas que no estaban en el planeamiento inicial - como era de esperar, es algo que ocurre en todo proyecto. Esto ha alterado un poco los tiempos, pero

en general, midiendo en *tiempos absolutos*, la estimación incial no iba muy desencaminada. Como durante la planificación del proyecto comenté podría pasar, la vertiente de demo de videojuego la dejé de lado, muy a mi pesar. Pero he preferido centrarme en explorar modelos, e investigar algo más profundamente.
En general, he seguido lo descrito en el documento de planificación del proyecto que presenté en una primera instancia; en cuanto a tecnologías, algoritmos e incluso procesos.

Distribución de horas

Me he organicé de manera previa al TFG para no tener que trabajar durante este cuatrimestre y dedicarme al proyecto principalmente. Así que he invertido bastante tiempo.

Veamos una tabla con horas, de forma orientativa:

Área	**Horas**	**%**
Arquitectura y motor de simulación	90	13.7 %
Modelado computacional (componentes biológicos)	90	13.7 %
Algoritmos genéticos y evolución	70	10.7 %
Reinforcement Learning (agentes fauna/clima)	95	14.5 %
Sistema neurosimbólico y forecasting	80	12.2 %
Visualización y UX/UI	41	6.2 %
Testing, debugging y optimización	60	9.1 %
Redacción de memoria	130	19.8 %
Total	**656**	**100 %**

Tabla 2: Distribución horas por área

Como se puede observar, he dedicado mucho tiempo a la memoria, incluyo desde generación de métricas y experimentos con el simulador, preparar diagramas, tablas...etc, a parte de la planificación para a intentar producir en orden coherente todo el contenido, así como sin *dependencias no resueltas* en cuanto a a contextos y conceptos (son muchos conceptos, y sistemas y he tenido que tener muy en cuenta que las cosas se cuenten en orden). También he tenido que realizar múltiples, múltiples y múltiples iteraciones para reducir y comprimir la extensión de la memoria drásticamente, y ésto ha incrementado los tiempos. En algunos casos he tenido que hacer ingeniería de compresión de textos. Pido una vez más disculpas a la lectora o lector.

También agregar que, en cuanto a arquitectura, es muy posible que haya sido más tiempo, ya que durante la implementación de otros sistemas la he ido extendiendo e incluso refactorizando cuando lo consideraba oportuno.

Y ahora el desglose - muy simplificado - de los sistemas principales. Puede verse el desarrollo en el historial de commits para más información.

Arquitectura y motor (90 h – 13.7 %)

- Refactorizaciones, componentes, entidades
- Sistema de eventos, bootstrap, sistemas de configuración
- Multihilo, snapshots, integración de InfluxDB
- Generación procedural de mapas, sistemas de hábitat

Modelado computacional (90 h – 13.7 %)

- Investigación
- Componentes y gestores, enfoque iterativo e incremental
- Handlers
- Sistemas de dormancia, gestión de energía
- Vectorización de componentes

Algoritmos Genéticos (70 h – 10.7 %)

- Investigación
- Integración con DEAP, base de algoritmos genéticos
- Ciclos de evolución, cálculo de fitness
- Operadores de mutación (adaptive, gaussian)
- Sistemas de control de población

Reinforcement Learning (95 h – 14.5 %)

- Investigación
- Agente RL para fauna
- Agentes RL para clima
- Adaptadores
- Múltiples arquitecturas de modelos y bucles de entrenamiento

Sistema Neurosimbólico (80 h – 12.2 %)

- Investigación
- Agente Equilibrium
- Módulos neurales, con sus arquitecturas etc.
- Módulos simbólicos
- Integración de módulos simbólicos
- Soporte para intervenciones

Figura 1: Commits

(El código inicial que se puede ver durante 2024, simplemente era la base para sistemas de escenas etc, de cara a la implementación de una posible demo de videojuego - viajar al origen de los tiempos)

Tablas y datos complementarios

Componentes y sus atributos

Expongo un mapeo general, de los atributos más relevantes de cada componente:

Componente/Handler	Atributos Gestionados
VitalComponent	▪ vitality ▪ max_vitality ▪ age ▪ biological_age ▪ aging_rate ▪ health_modifier ▪ somatic_integrity ▪ max_somatic_integrity ▪ integrity_regeneration_rate ▪ accumulated_decay ▪ accumulated_stress ▪ vitality_history

Componente/Handler	Atributos Gestionados
GrowthComponent	▪ growth_stage ▪ current_size ▪ max_size ▪ growth_modifier ▪ growth_efficiency ▪ initial_size ▪ total_stages ▪ stage_thresholds
PhotosyntheticMetabolismComp.	▪ photosynthesis_efficiency ▪ base_photosynthesis_efficiency ▪ respiration_rate ▪ base_respiration_rate ▪ metabolic_activity ▪ light_availability ▪ temperature_modifier ▪ water_modifier ▪ optimal_respiration_ratio ▪ energy_factor
AutotrophicNutritionComponent	▪ nutrient_absorption_rate ▪ mycorrhizal_rate ▪ base_nutritive_value ▪ current_nutritive_value ▪ base_toxicity ▪ current_toxicity

Componente/Handler	Atributos Gestionados
HeterotrophicNutritionComponent	▪ hunger_level ▪ thirst_level ▪ hunger_rate ▪ thirst_rate ▪ metabolism_efficiency
WeatherAdaptationComponent	▪ cold_resistance ▪ heat_resistance ▪ optimal_temperature ▪ base_sigma ▪ max_stress_delta ▪ temperature_exposure_history
MovementComponent	▪ movement_energy_cost ▪ current_position ▪ can_move ▪ movement_valid
TransformComponent	▪ x ▪ y
StressHandler	▪ stress_level ▪ max_stress ▪ lifespan
EnergyHandler	▪ energy_reserves ▪ max_energy_reserves

Métricas Ecológicas

Para poder llevar a cabo todo el proceso, hay muchos sistemas involucrados, desde sistemas de métricas y análisis, recolectores de datos etc. Pero los principales que aquí serán de utilidad, son el sistema de contribuyentes y el de cálculo del score de bioma (links a código).

Balance poblacional

Con este contribuyente, evalúo el equilibrio entre flora y fauna - considero como proporción ideal el 60 % de flora:

$$R_{flora} = \frac{N_{flora}}{N_{total}} \tag{1}$$

$$S_{balance} = \begin{cases} 1{,}0 - \text{mín}\left(1{,}0, \frac{R_{flora} - R_{ideal}}{1{,}0 - R_{ideal}} \cdot 0{,}8\right), & \text{si } R_{flora} > R_{ideal} \\ \text{máx}\left(0{,}0, \frac{R_{flora}}{R_{ideal}}\right), & \text{si } R_{flora} \leq R_{ideal} \end{cases} \tag{2}$$

Símbolo	Descripción
N_{flora}	Número de entidades flora.
N_{fauna}	Número de entidades fauna.
N_{total}	Población total ($N_{flora} + N_{fauna}$).
R_{flora}	Proporción de flora.
R_{ideal}	Proporción ideal de flora (0.6).
$S_{balance}$	Puntuación de balance poblacional.

Toxicidad ambiental

Menor toxicidad - mejor puntuación:

$$S_{toxicity} = \text{máx}\left(0{,}0, 1{,}0 - \frac{T_{avg}}{T_{critical}}\right) \tag{3}$$

Símbolo	Descripción
T_{avg}	Toxicidad media del bioma.
$T_{critical}$	Umbral crítico de toxicidad (50.0, la mmitad).
$S_{toxicity}$	Puntuación de toxicidad normalizada.

Condiciones climáticas

Evalúo el clima mediante 3 factores - temperatura, humedad, precipitación - cada uno con rangos óptimos específicos por tipo de bioma:

$$S_{factor} = \begin{cases} 1{,}0, & \text{si } V_{min} \leq V \leq V_{max} \\ \text{máx}\left(0{,}0, 1{,}0 - \frac{V_{min} - V}{V_{min}}\right), & \text{si } V < V_{min} \\ \text{máx}\left(0{,}0, 1{,}0 - \frac{V - V_{max}}{100{,}0 - V_{max}}\right), & \text{si } V > V_{max} \end{cases} \tag{4}$$

$$S_{climate} = \frac{\sum_i S_{factor,i} \cdot W_i}{\sum_i W_i} \tag{5}$$

Símbolo	Descripción
V	Valor del factor climático.
V_{min}, V_{max}	Rango óptimo del factor.
S_{factor}	Puntuación del factor individual.
W_i	Peso del factor ($W_{temp} = 0{,}4$, $W_{hum} = 0{,}3$, $W_{prec} = 0{,}3$).
$S_{climate}$	Puntuación climática final.

Biodiversidad y equilibrio ecológico

Aquí combino diversidad de especies y equilibrio depredador-presa. En general tendré disponible el índice de Shannon-Wiener para todo escenario, pero ha habido veces que en la simulación fallaba, con lo que, por si acaso he implementado un método básico de fallback. No obstante, también considero equilibrio en población y, con estas dos puntuaciones, doy score final de biodiversidad:

$$N_{flora,est} = \text{mín}(5, \text{máx}(1, \lfloor N_{flora}/3 \rfloor)) \tag{6}$$

$$N_{fauna,est} = \text{mín}(4, \text{máx}(1, \lfloor N_{fauna}/2 \rfloor)) \tag{7}$$

$$P_{penalty} = \begin{cases} 0{,}7, & \text{si } N_{species,total} < 3 \\ 1{,}0, & \text{en otro caso} \end{cases} \tag{8}$$

Símbolo	Descripción
$I_{Shannon}$	Índice Shannon-Wiener.
$N_{flora,est}$	Especies de flora estimadas.
$N_{fauna,est}$	Especies de fauna estimadas.
$N_{species,total}$	Total de especies estimadas.
$P_{penalty}$	Factor de penalización por baja diversidad.
$N_{species}$	Número total de especies.
$N_{predators}$	Población de depredadores.
N_{prey}	Población de presas.
R_{pred}	Ratio depredador-presa.
R_{min}	Ratio mínimo óptimo (0.1).
R_{max}	Ratio máximo óptimo (0.3).
R_{mid}	Ratio medio óptimo (0.2).
$S_{species}$	Puntuación de diversidad.
$S_{balance}$	Puntuación de equilibrio.

$$S_{species} = \begin{cases} I_{Shannon}, & \text{si está disponible} \\ \text{mín}\left(1{,}0, \frac{N_{flora,est}+N_{fauna,est}}{8{,}0}\right) \cdot P_{penalty}, & \text{en otro caso} \end{cases} \tag{9}$$

$$R_{pred} = \frac{N_{predators}}{N_{prey}} \tag{10}$$

$$S_{balance} = \begin{cases} 0{,}1, & \text{si } R_{pred} = \infty \\ 0{,}4, & \text{si } R_{pred} = 0 \text{ y } N_{prey} > 0 \\ 0{,}4 + \frac{R_{pred}}{R_{min}} \cdot 0{,}6, & \text{si } R_{pred} < R_{min} \\ 1{,}0 - \text{mín}\left(1{,}0, \frac{R_{pred}-R_{max}}{2{,}0}\right) \cdot 0{,}8, & \text{si } R_{pred} > R_{max} \\ 1{,}0 - \frac{|R_{pred}-R_{mid}|}{R_{max}-R_{min}} \cdot 0{,}2, & \text{en otro caso} \end{cases} \tag{11}$$

$$S_{biodiversity} = S_{species} \cdot 0{,}6 + S_{balance} \cdot 0{,}4 \tag{12}$$

Salud del ecosistema

Para la salud simplemente calculo la media ponderada de factores ambientales:

$$F_{stress} = 1{,}0 - \frac{\text{mín}(100{,}0, L_{stress})}{100{,}0} \tag{13}$$

Símbolo	Descripción
L_{stress}	Nivel promedio de estrés.
S_{avg}	Tamaño promedio de entidades.
$A_{climate}$	Factor de adaptación climática.
$B_{balance}$	Factor de balance de especies.
F_{stress}	Factor de estrés normalizado.
F_{size}	Factor de tamaño normalizado.
$F_{toxicity}$	Factor de toxicidad normalizado.
W_{total}	Suma de pesos (5.3).

$$F_{size} = \begin{cases} \text{mín}\left(1{,}0, \frac{S_{avg}}{2{,}0}\right), & \text{si } S_{avg} \leq 2{,}0 \\ \text{máx}\left(0{,}0, 1{,}0 - \frac{S_{avg}-2{,}0}{3{,}0}\right), & \text{si } S_{avg} > 2{,}0 \end{cases} \tag{14}$$

$$F_{toxicity} = 1{,}0 - \frac{\text{mín}(100{,}0, T_{avg})}{100{,}0} \tag{15}$$

$$S_{health} = \frac{F_{stress} \cdot 1{,}5 + F_{size} \cdot 1{,}0 + A_{climate} \cdot 1{,}0 + B_{balance} \cdot 1{,}0 + F_{toxicity} \cdot 0{,}8}{W_{total}} \tag{16}$$

Cálculo de la puntuación final del bioma

Para terminar, suma ponderada de contribuyentes:

$$S_{weighted} = \sum_i S_i \cdot W_i \tag{17}$$

$$S_{normalized} = \frac{S_{weighted}}{\sum_i W_i} \tag{18}$$

$$S_{final} = S_{normalized} \times 10{,}0 \tag{19}$$

Símbolo	Descripción
S_i	Puntuación del contribuyente i.
W_i	Peso del contribuyente i.
$S_{weighted}$	Suma ponderada total.
$S_{normalized}$	Puntuación normalizada [0,1].
S_{final}	Puntuación final [0,10].
Q_{biome}	Calidad categórica del bioma.

$$Q_{biome} = \begin{cases} \text{CRITICAL}, & \text{si } S_{normalized} < 0{,}2 \\ \text{UNSTABLE}, & \text{si } 0{,}2 \leq S_{normalized} < 0{,}4 \\ \text{MODERATE}, & \text{si } 0{,}4 \leq S_{normalized} < 0{,}6 \\ \text{HEALTHY}, & \text{si } 0{,}6 \leq S_{normalized} < 0{,}8 \\ \text{EDEN}, & \text{si } S_{normalized} \geq 0{,}8 \end{cases} \tag{20}$$

Los pesos por defecto que en las pruebas me han parecido apropiados son: Balance poblacional ($W = 1{,}1$), Clima ($W = 1{,}0$), Biodiversidad ($W = 1{,}2$) y Salud del ecosistema ($W = 1{,}0$).

Evolución - fitness de entidades

Expongo aquí de forma formulaica, lo que he querido expresar con las funciones de fitness en el proyecto - ver código.

Función de fitness para flora

Para evaluación, combino seis componentes principales:

$$F_{flora} = F_{temp} + F_{hum} + F_{prec} + F_{surv} + F_{energy} + F_{nutr} - P_{constraints} \tag{21}$$

Símbolo	Descripción
F_{flora}	Fitness total de flora.
F_{temp}	Componente de adaptación térmica.
F_{hum}	Componente de adaptación humedad.
F_{prec}	Componente de adaptación precipitación.
F_{surv}	Componente de supervivencia general.
F_{energy}	Componente de eficiencia energética.
F_{nutr}	Componente nutricional.
$P_{constraints}$	Penalizaciones por restricciones.
T_{avg}	Temperatura promedio.
T_{opt}	Temperatura óptima.
σ	Desviación estándar térmica.
R	Resistencia al frío/calor.
ϵ	Factor de corrección pequeño.

$$F_{temp} = 5{,}0 \cdot \exp\left(-\frac{(T_{avg} - T_{opt})^2}{2\sigma^2}\right) \tag{22}$$

$$\sigma = \frac{15{,}0}{1{,}0 - R + \epsilon} \tag{23}$$

$$F_{hum} = \begin{cases} 5{,}0(1 - R_{resp}) + 2{,}0P_{eff} & \text{si } H_{avg} < 30\,\% \\ 3{,}0M_{act} + 3{,}0P_{eff} & \text{si } H_{avg} > 70\,\% \\ 3{,}0 & \text{caso intermedio} \end{cases} \tag{24}$$

Símbolo	Descripción
H_{avg}	Humedad promedio.
R_{resp}	Tasa de respiración.
P_{eff}	Eficiencia fotosintética efectiva.
M_{act}	Actividad metabólica.
P_{avg}	Precipitación promedio.
E_{res}	Reservas energéticas.
G_{mod}	Modificador de crecimiento.
S_{max}	Tamaño máximo.
V_{max}	Vitalidad máxima.
A_{rate}	Tasa de envejecimiento.
G_{eff}	Eficiencia de crecimiento.
H_{mod}	Modificador de hidratación.
M_{opt}	Metabolismo óptimo.

$$F_{prec} = \begin{cases} 4{,}0P_{eff} + 3{,}0\frac{E_{res}}{200} & \text{si } P_{avg} < 30\text{mm} \\ 3{,}0G_{mod} + 2{,}0\frac{S_{max}}{5} & \text{si } P_{avg} > 100\text{mm} \\ 3{,}0 & \text{caso intermedio} \end{cases} \tag{25}$$

$$F_{surv} = 2{,}0\frac{V_{max}}{200} + 1{,}5\left(1 - \frac{A_{rate}}{2}\right) \tag{26}$$

$$F_{energy} = 2{,}0G_{eff} + 1{,}5H_{mod} + 2{,}0M_{opt} - 10{,}0\,\text{máx}(0, R_{resp} - 0{,}1) \tag{27}$$

$$M_{opt} = 1{,}0 - \frac{|M_{act} - 0{,}8|}{0{,}8} \tag{28}$$

$$F_{nutr} = 3{,}0N_{abs}I(H_{avg} > 60\,\%) + 2{,}5M_{rate} \cdot 10 \cdot I(P_{avg} < 50\text{mm}) + T_{def} \tag{29}$$

Símbolo	Descripción
N_{abs}	Absorción de nutrientes.
M_{rate}	Tasa micorrízica.
T_{def}	Defensa contra toxicidad.
T_{tox}	Toxicidad base.
$I(\cdot)$	Función indicadora.
P_{base}	Fotosíntesis base.
L_{disp}	Disponibilidad lumínica (0.5).
T_{mod}	Modificador térmico (0.7).
W_{mod}	Modificador hídrico (0.6).
R_{eff}	Respiración efectiva.
R_{base}	Respiración base.

$$T_{def} = \begin{cases} 1{,}0T_{tox} & \text{si } T_{tox} < 0{,}4 \\ -2{,}0(T_{tox} - 0{,}4) & \text{si } T_{tox} \geq 0{,}4 \end{cases} \tag{30}$$

$$P_{eff} = P_{base} \cdot L_{disp} \cdot T_{mod} \cdot W_{mod} \cdot M_{act} \tag{31}$$

$$R_{eff} = R_{base} \cdot M_{act} \tag{32}$$

Penalizaciones energéticas y por valores extremos:

$$\text{Si } P_{eff} \leq R_{eff}: \quad F_{total} \leftarrow F_{total} \times 0{,}1 \tag{33}$$

$$\text{Si } \frac{P_{eff}}{R_{eff}} > 1{,}0: \quad F_{total} \leftarrow F_{total} + \text{mín}\left(5{,}0, \frac{P_{eff}}{R_{eff}} - 1{,}0\right) \times 2{,}0 \tag{34}$$

$$P_{foto\_excesiva} = 10{,}0(e^{4(P_{base} - 0{,}7)} - 1) \quad \text{si } P_{base} > 0{,}7 \tag{35}$$

$$P_{ratio\_extremo} = 20{,}0\left(\frac{P_{base}}{R_{base}} - 3{,}0\right) \quad \text{si } \frac{P_{base}}{R_{base}} > 3{,}0 \tag{36}$$

$$P_{resp\_insuf} = 30{,}0(0{,}18 - R_{base}) \quad \text{si } R_{base} < 0{,}18 \tag{37}$$

$$P_{resp\_biologica} = 80{,}0(0{,}35 \times P_{base} - R_{base}) \quad \text{si } R_{base} < 0{,}35 \times P_{base} \tag{38}$$

Función de fitness para fauna

Aquí sustituyo componentes fotosintéticos por adaptaciones heterótrofas:

$$F_{fauna} = F_{temp} + F_{nutr} + F_{energy} + F_{health} + F_{climate} - P_{constraints} \tag{39}$$

Símbolo	Descripción
F_{fauna}	Fitness total de fauna.
F_{health}	Componente de salud y vitalidad.
$F_{climate}$	Adaptación climática específica.
H_{opt}	Optimización de hambre.
T_{opt}	Optimización de sed.
H_{rate}	Tasa de hambre.
T_{rate}	Tasa de sed.

$$F_{temp} = 4{,}0 \cdot \exp\left(-\frac{(T_{avg} - T_{opt})^2}{2\sigma^2}\right) \tag{40}$$

$$F_{nutr} = 2{,}0H_{opt} + 2{,}0T_{opt} \tag{41}$$

$$H_{opt} = \begin{cases} 1{,}0 - \frac{H_{rate} - 1{,}1}{0{,}3} \times 0{,}5 & \text{si } H_{rate} \leq 1{,}1 \\ 1{,}0 - \frac{H_{rate} - 1{,}1}{0{,}4} \times 2{,}0 & \text{si } H_{rate} > 1{,}1 \end{cases} \tag{42}$$

$$T_{opt} = \begin{cases} 1{,}0 - \frac{T_{rate} - 1{,}2}{0{,}8} \times 2{,}0 & \text{si } H_{avg} < 40\,\% \\ T_{opt\_humedo} & \text{si } H_{avg} \geq 40\,\% \end{cases} \tag{43}$$

$$T_{opt\_humedo} = \begin{cases} 1{,}0 - \frac{1{,}5 - T_{rate}}{0{,}3} \times 0{,}5 & \text{si } T_{rate} \leq 1{,}5 \\ 1{,}0 - \frac{T_{rate} - 1{,}5}{0{,}5} \times 1{,}5 & \text{si } T_{rate} > 1{,}5 \end{cases} \tag{44}$$

Símbolo	Descripción
T_{opt_humedo}	Optimización sed en ambiente húmedo.
E_{ideal}	Reservas energéticas ideales.

$$F_{energy} = 2{,}5\,\text{máx}\left(0, 1 - \frac{|E_{res} - E_{ideal}|}{E_{ideal}}\right) \tag{45}$$

$$E_{ideal} = 100 + S_{max} \times 20 \tag{46}$$

$$F_{health} = 2{,}0\frac{V_{max}}{200} + 1{,}5\left(1 - \frac{A_{rate}}{2}\right) \tag{47}$$

$$\begin{aligned} F_{climate} = {} & 1{,}5 \cdot I(P_{avg} > 80 \wedge R_{cold} > 0{,}3) \\ & + 1{,}5 \cdot I(P_{avg} < 20 \wedge R_{heat} > 0{,}3) \\ & + 1{,}0 \cdot I(T_{avg} < 5 \wedge S_{max} > 3{,}0) \\ & + 1{,}0 \cdot I(T_{avg} > 30 \wedge S_{max} < 2{,}0) \end{aligned} \tag{48}$$

Símbolo	Descripción
R_{cold}	Resistencia al frío.
R_{heat}	Resistencia al calor.
F_{base}	Fitness base de supervivencia.
M_{eff}	Eficiencia metabólica.

$$F_{base} = 3{,}5 \times M_{eff} \tag{49}$$

$$F_{total} \leftarrow \text{máx}(F_{total}, F_{base}) \tag{50}$$

Penalizo linealmente y agrego factores severos forma severa:

$$P_{hunger} = 5{,}0 \times \frac{H_{rate} - 0{,}8}{0{,}7} \tag{51}$$

$$P_{thirst} = 5{,}0 \times \frac{T_{rate} - 1{,}2}{0{,}8} \tag{52}$$

Símbolo	Descripción
P_{hunger}	Penalización linear por hambre.
P_{thirst}	Penalización linear por sed.
P_{hunger_severa}	Penalización severa por hambre.
P_{thirst_severa}	Penalización severa por sed.

$$P_{hunger\_severa} = 8{,}0 \times (H_{rate} - 1{,}0) \quad \text{si } H_{rate} > 1{,}0 \tag{53}$$

$$P_{thirst\_severa} = 8{,}0 \times (T_{rate} - 1{,}4) \quad \text{si } T_{rate} > 1{,}4 \tag{54}$$

Continuidad de ventanas deslizantes

Tabla 4: Ventanas deslizantes, resutados - sequencias 0 y 1

Sequence 0

VENTANA ACTUAL:
Índices: 0 a 14
Snapshot IDs: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]

Datos de características (ventana actual resumida):

ID	Prey	Pred.	Flora	Eco.Stab.	Stress
0	35	5	66	0.891	0.005
1	35	5	57	0.895	0.019
2	33	5	51	0.892	0.060
3	33	5	44	0.882	0.104
4	32	5	41	0.877	0.156
5	29	5	38	0.883	0.202
6	26	5	37	0.887	0.242
7	25	5	35	0.882	0.290
8	22	5	34	0.877	0.337
9	19	5	33	0.886	0.359
10	18	5	33	0.877	0.374
11	16	5	32	0.886	0.397
12	14	5	32	0.876	0.431
13	14	5	32	0.876	0.487
14	14	5	32	0.876	0.550

TARGETS:
Índices: 15 a 19
Snapshot IDs: [15, 16, 17, 18, 19]

Datos de objetivos:

ID	Prey	Pred.	Flora	Eco.Stab.	Stress
15	13	5	32	0.868	0.608
16	12	5	32	0.865	0.643
17	8	5	33	0.833	0.602
18	8	5	32	0.836	1.071
19	7	5	31	0.828	3.656

SIGUIENTE VENTANA (Secuencia 1, Simulación 0):
Índices: 1 a 15
Snapshot IDs: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]

CONTINUIDAD:
Ventanas consecutivas: 0 + 1 → 1
Solapamiento entre targets y siguiente ventana: 1 índices
Snapshot IDs solapados: [15]

Sequence 1

VENTANA ACTUAL:
Índices: 1 a 15
Snapshot IDs: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]

Datos de características (ventana actual resumida):

ID	Prey	Pred.	Flora	Eco.Stab.	Stress
1	35	5	57	0.895	0.019
2	33	5	51	0.892	0.060
3	33	5	44	0.882	0.104
4	32	5	41	0.877	0.156
5	29	5	38	0.883	0.202
6	26	5	37	0.887	0.242
7	25	5	35	0.882	0.290
8	22	5	34	0.877	0.337
9	19	5	33	0.886	0.359
10	18	5	33	0.877	0.374
11	16	5	32	0.886	0.397
12	14	5	32	0.876	0.431
13	14	5	32	0.876	0.487
14	14	5	32	0.876	0.550
15	13	5	32	0.868	0.608

TARGETS:
Índices: 16 a 20
Snapshot IDs: [16, 17, 18, 19, 20]

Datos de objetivos:

ID	Prey	Pred.	Flora	Eco.Stab.	Stress
16	12	5	32	0.865	0.643
17	8	5	33	0.833	0.602
18	8	5	32	0.836	1.071
19	7	5	31	0.828	3.656
20	6	5	31	0.814	5.611

SIGUIENTE VENTANA (Secuencia 2, Simulación 0):
Índices: 2 a 16
Snapshot IDs: [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]

CONTINUIDAD:
Ventanas consecutivas: 1 + 1 → 2
Solapamiento entre targets y siguiente ventana: 1 índices
Snapshot IDs solapados: [16]

Datos expandidos

Expongo las mediciones completas de las ventanas referenciadas:

Tabla 5: Datos completos de ventana - Sequence 0

ID	Prey	Pred.	Flora	Temp.	Humid.	Precip.	CO_2	Bio.Index	Bio.Density	Eco.Stab.	Biome	Env.Press.	Avg.Stress
0	35	5	66	-10.11	34.50	15.85	379.51	0.940	0.282	0.891	8.43	0.323	0.005
1	35	5	57	-6.51	37.07	26.51	373.65	0.944	0.273	0.895	8.45	0.295	0.019
2	33	5	51	-4.76	41.17	34.84	368.96	0.946	0.269	0.892	8.38	0.298	0.060
3	33	5	44	-3.70	42.50	39.19	364.81	0.944	0.272	0.882	8.20	0.309	0.104
4	32	5	41	-2.95	43.35	42.31	361.05	0.940	0.272	0.877	8.14	0.327	0.156
5	29	5	38	-2.50	43.89	44.73	357.64	0.949	0.272	0.883	8.17	0.346	0.202
6	26	5	37	-1.29	45.92	51.14	354.37	0.949	0.278	0.887	8.25	0.354	0.242
7	25	5	35	-0.98	47.31	55.75	351.23	0.942	0.279	0.882	8.22	0.375	0.290
8	22	5	34	-1.02	47.02	58.48	348.14	0.930	0.279	0.877	8.24	0.399	0.337
9	19	5	33	-1.23	47.30	61.70	345.14	0.941	0.283	0.886	8.31	0.412	0.359
10	18	5	33	-3.77	49.40	76.60	342.16	0.920	0.283	0.877	8.34	0.445	0.374
11	16	5	32	-2.05	47.40	77.90	339.19	0.927	0.287	0.886	8.45	0.439	0.397
12	14	5	32	-1.32	46.54	77.49	336.24	0.924	0.287	0.876	8.28	0.449	0.431
13	14	5	32	-1.35	46.65	75.73	333.29	0.924	0.287	0.876	8.28	0.477	0.487
14	14	5	32	-1.37	46.65	73.46	330.34	0.924	0.287	0.876	8.28	0.509	0.550

Tabla 6: Datos completos de ventana - Sequence 1

ID	Prey	Pred.	Flora	Temp.	Humid.	Precip.	CO_2	Bio.Index	Bio.Density	Eco.Stab.	Biome	Env.Press.	Avg.Stress
1	35	5	57	-6.51	37.07	26.51	373.65	0.944	0.273	0.895	8.45	0.295	0.019
2	33	5	51	-4.76	41.17	34.84	368.96	0.946	0.269	0.892	8.38	0.298	0.060
3	33	5	44	-3.70	42.50	39.19	364.81	0.944	0.272	0.882	8.20	0.309	0.104
4	32	5	41	-2.95	43.35	42.31	361.05	0.940	0.272	0.877	8.14	0.327	0.156
5	29	5	38	-2.50	43.89	44.73	357.64	0.949	0.272	0.883	8.17	0.346	0.202
6	26	5	37	-1.29	45.92	51.14	354.37	0.949	0.278	0.887	8.25	0.354	0.242
7	25	5	35	-0.98	47.31	55.75	351.23	0.942	0.279	0.882	8.22	0.375	0.290
8	22	5	34	-1.02	47.02	58.48	348.14	0.930	0.279	0.877	8.24	0.399	0.337
9	19	5	33	-1.23	47.30	61.70	345.14	0.941	0.283	0.886	8.31	0.412	0.359
10	18	5	33	-3.77	49.40	76.60	342.16	0.920	0.283	0.877	8.34	0.445	0.374
11	16	5	32	-2.05	47.40	77.90	339.19	0.927	0.287	0.886	8.45	0.439	0.397
12	14	5	32	-1.32	46.54	77.49	336.24	0.924	0.287	0.876	8.28	0.449	0.431
13	14	5	32	-1.35	46.65	75.73	333.29	0.924	0.287	0.876	8.28	0.477	0.487
14	14	5	32	-1.37	46.65	73.46	330.34	0.924	0.287	0.876	8.28	0.509	0.550
15	13	5	32	-1.27	46.29	72.17	327.41	0.917	0.287	0.868	8.19	0.537	0.608

Estadísticos

En la tabla 7 muestro un resúmen con los estadísticos de la simulación a la que pertenecen las ventanas mostradas:

Tabla 7: Resúmen de estadísticos

Metric	Mean	Std Dev	Min	Max
Prey Population	23.13	8.92	14	35
Predator Population	5.00	0.00	5	5
Flora Count	39.80	10.15	32	66
Temperature	-2.91	2.40	-10.11	-0.98
Humidity	44.68	3.72	34.50	49.40
Precipitation	53.78	20.36	15.85	77.90
CO_2 Level	353.71	16.07	330.34	379.51
Ecosystem Stability	0.883	0.007	0.868	0.895
Average Stress	0.301	0.178	0.005	0.608

Continuidad en ventana vs salto en horizonte de una característica

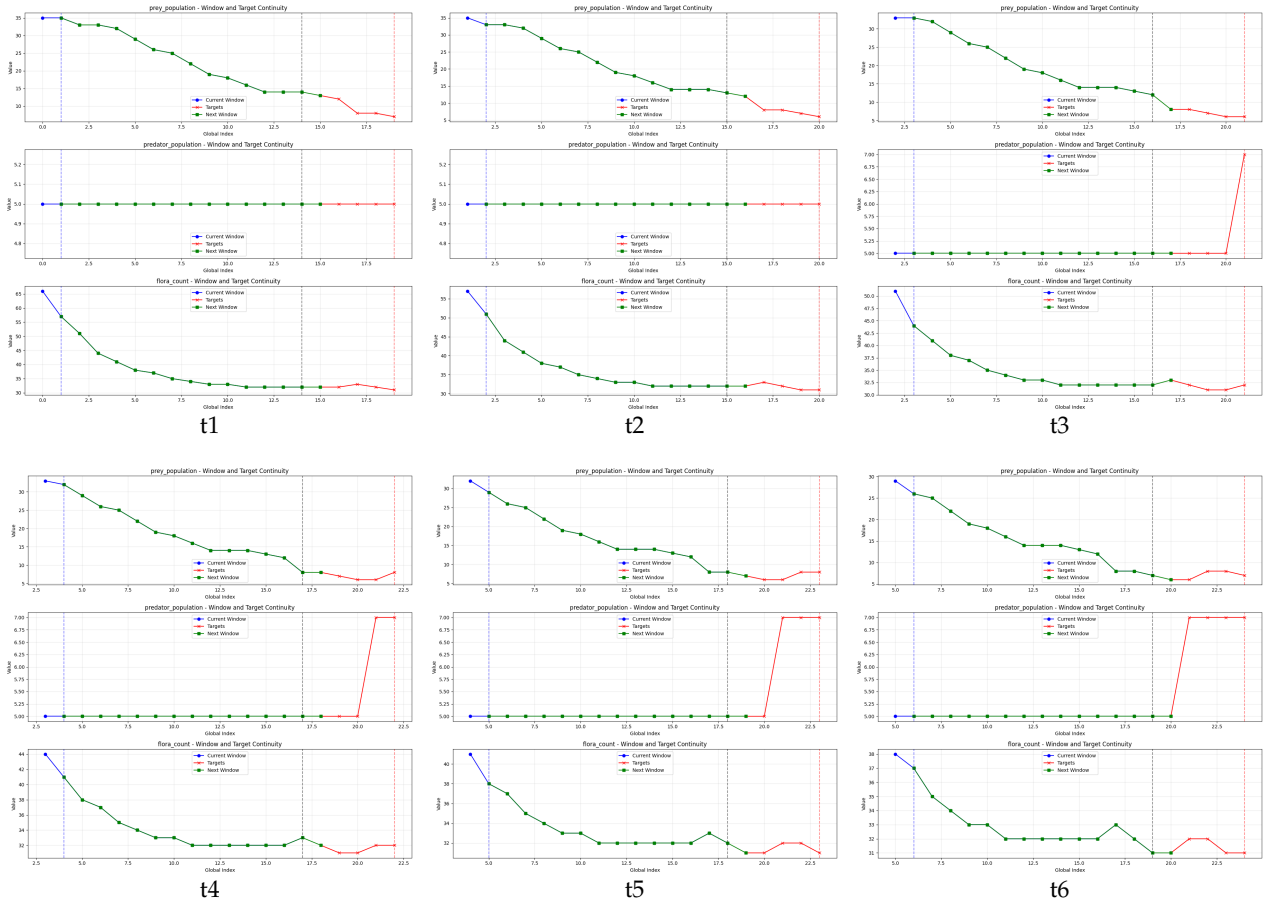


Figura 2: Continuidad en ventana de input vs saltos inesperados (en una feature) en el horizonte de predicción para seis instantes (t1–t6)

Comparativa de puntos clave de PPO/DQN en la simullación

Punto-clave	Cómo se materializa en mi entorno	Implicaciones para la *elección*
Valor vs Actor–Critic	Acciones discretas ($k \approx 16$); mañana podría añadir +50 eventos.	*DQN*: red sample-efficient para k pequeño. *PPO/A2C*: sólo agrandan la capa soft-max si k crece.
On-policy vs Off-policy	No hay dataset fijo, cada episodio lo creo al momento generando datos *on-the-fly*, sin coste; las estaciones cambian lento.	Off-policy exprime experiencias antiguas; On-policy (PPO) mantiene la política estable si toco reward o entorno. Si hubiera un dataset fijo, se usarían métodos de offline-RL
Exploración	16 acciones discretas — la exploración aleatoria cubre todo el espacio con bajo coste.	**DQN**: usa ε-greedy; **PPO/A2C**: controla la entropía con `ent_coef`.
No-estacionariedad	Con cada estación, se cambian las distribuciones; normalizo, y la variable `season` está en la observación.	Clipping de PPO ayuda a no *volcar* la política cuando aparecen transiciones bruscas. Con DQN, puedo usar RNN para memoria histórica (SB3 tiene soporte, muy sencillo)
Horizonte largo	Un episodio dura 365 pasos; recompensa densa. Me interesa que aprenda a planificar el agente.	Con un factor descuento $\gamma \approx 0{,}97 - 0{,}99$ dejo que el algortimo *mire* varias semanas sin perder señal inmediata; No influye si PPO/DQN
Coste de simulación	Simular un año tarda milisegundos.	Puedo permitirme `n_steps`=2048 en PPO o buffers de 1 M de tuplas en DQN sin preocuparme.

Tabla 8: Resumen de criterios que influyen en la selección del algoritmo de RL. La tabla compara *qué piden los datos* y *qué ofrecen* DQN, PPO y A2C sin declarar un ganador único; mi elección final se fundamenta en las pruebas empíricas mostradas en la Fig. 3.23.

Reglas del módulo simbólico

Tabla 9: Reglas del sistema simbólico basado en reglas (versión alineada con `RuleBasedSymbolicModule`)

Categoría	Condición	Resultado
Estado de especies	population < 7	species_status $\leftarrow$ ENDANGERED species_action $\leftarrow$ PROTECTION_NEEDED
	avg_stress > 75	species_status $\leftarrow$ STRESSED species_action $\leftarrow$ STRESS_REDUCTION_NEEDED
	population $> 40 \wedge$ type $=$ fauna	species_status $\leftarrow$ OVERPOPULATED species_action $\leftarrow$ POPULATION_CONTROL_NEEDED
Dinámica depredador–presa	predator_prey_ratio $> 0{,}3$	predator_prey_balance $\leftarrow$ PREDATOR_DOMINANT ecosystem_risk $\leftarrow$ PREY_EXTINCTION_RISK recommended_action $\leftarrow$ REDUCE_PRED_PRESSURE
	predator_prey_ratio $< 0{,}1 \wedge$ prey_population $>$ 15	predator_prey_balance $\leftarrow$ PREY_DOMINANT ecosystem_risk $\leftarrow$ OVERPOPULATION_RISK recommended_action $\leftarrow$ INCREASE_PRED_PRESSURE
Estabilidad del ecosistema	biodiversity_index $< 0{,}3$	biodiversity_status $\leftarrow$ CRITICAL recommended_action $\leftarrow$ INTRODUCE_SPECIES_DIVERSITY
	ecosystem_stability $< 0{,}4$	stability_status $\leftarrow$ UNSTABLE recommended_action $\leftarrow$ ECOSYSTEM_INTERVENTION_NEEDED
	ecosystem_stability $> 0{,}8$	stability_status $\leftarrow$ HIGHLY_STABLE intervention_priority $\leftarrow$ LOW

Métricas devueltas por GraphBasedSymbolicModule

Tabla 10: Métricas devueltas por `GraphBasedSymbolicModule`

Métrica	Cómo se calcula	Propósito/Objetivo	Ejemplo/Impacto
Centralidad de grado (`degree_centrality`)	Se cuentan las conexiones directas de cada especie y se normalizan por el total posible.	Identifica especies-*hub* con numerosas interacciones en el ecosistema.	Cuando los robles (`oak_tree`) tienen alta centralidad, el sistema prioriza su protección durante sequías - alimentan a múltiples herbívoros como ciervos (`deer`), etc.
Centralidad de intermediación (`betweenness_centrality`)	Esta métrica analiza qué fracción de rutas mínimas entre pares atraviesa cada especie.	Busca especies que funcionen como puentes entre comunidades separadas.	El zorro (`arctic_fox`) en la tundra conecta diferentes niveles tróficos - caza pequeños herbívoros pero también es presa del oso polar. Claro, si se extingue rompería esta conexión, y aislaría poblaciones de otras especies de las que el oso se alimente.
Centralidad de Eigenvector (`eigenvector_centrality`)	Las conexiones se ponderan más si enlazan con especies que también están muy conectadas. Más improtancia, básicamente.	Me ayuda a revelar especies influyentes que podrían ser *keystone*.	Si la pantera (`panther`) muestra alta centralidad de vector propio, el módulo neurosimbólico puede activar `ADJUST_EVOLUTION_CYCLE` con factor `ACCELERATE_SIGNIFICANT` para protegerla.
Conectividad (`connectivity`)	Comparo la proporción de enlaces presentes frente a los posibles en el subgrafo de especies.	Cuantifica qué tan densamente conectada está la red.	En la sabana, los herbívoros tienen múltiples opciones de flora para alimentarse (conectividad 0.7), mientras que en el desierto dependen casi exclusivamente de cactus y palmeras datileras (conectividad 0.2). Esto hace que el desierto sea más vulnerable a la pérdida de especies vegetales.
Coeficiente de agrupamiento medio (`average_clustering`)	La densidad de triángulos (vecinos conectados entre sí) se promedia por especie.	Aporta información sobre el nivel típico de cohesión local en la red trófica.	Durante la estación lluviosa (`Season.SPRING`), el `average_clustering` del bioma de savana aumenta porque las congregaciones de herbívoros alrededor de fuentes de agua crean interacciones más densas (idealmente).
Recuento de comunidades (`community_count`)	Utilizo Greedy Modularity para agrupar y después cuento los clusters resultantes.	Estima cuántos nichos o grupos funcionales existen en el bioma.	En el desierto actual se detectan solo dos comunidades; el módulo neurosimbólico recomendaría `INTRODUCE_SPECIES_DIVERSITY`, y recomienda intervenciones para añadir especies como `desert_sage`.
Robustez (`robustness`)	La densidad se evalúa antes y después de eliminar aleatoriamente un 20 % de especies.	Mide la capacidad del sistema para resistir perturbaciones como extinciones.	Cuando simulo la pérdida aleatoria de especies en la taiga, las conexiones se mantienen relativamente estables (robustez 0.85) porque existe redundancia en roles ecológicos. Pero, en el desierto, sin embargo, perder incluso especies secundarias colapsa rápidamente la red (robustez 0.3) porque cada especie cumple un nicho único.
Nivel trófico (`trophic_levels`)	Los productores reciben valor 1; cada consumidor obtiene 1 + máx(nivel de sus presas).	Sitúa cada organismo en su posición dentro de la cadena alimentaria.	El lobo (`wolf`) en la taiga ocupa nivel 3, pero el oso polar (`polar_bear`) en la tundra solo alcanza nivel 2 - esto explica la menor estabilidad observada en este último ecosistema.
Longitud máxima de cadena (`food_chain_length`)	Tomo directamente el nivel trófico máximo que se haya calculado.	Da una idea clara sobre la complejidad vertical del ecosistema.	Como ejemplo, imaginemos que el ecosistema tropical tiene longitud 4 (musgo→insectos→ranas→jaguares) frente a 2 en la tundra. Cuando `_analyze_trophic_levels()` detecta este desequilibrio, genera intervenciones específicas para hacer más compljea la red trófica de la tundra.

Sistema de intervenciones neurosimbólicas

La integración y las intervenciones que genero en éste sistema, se pueden ver en éste fichero.

INTERVENCIONES DE ESPECIES EN RIESGO

Especie `ENDANGERED`

Intervención: `SPAWN_ENTITIES`
Parámetros: Spawn count calculado según dieta

Especie `STRESSED`

Intervención: `ADJUST_EVOLUTION_CYCLE`
Parámetros: Factor = máx(`ACCELERATE_EXTREME`, `ACCELERATE_SIG` + (1 − salud_eco) × 0,3)

Especie `OVERPOPULATED`

Intervención: `ADJUST_EVOLUTION_CYCLE`
Parámetros: Factor = `SLOW_EXTREME` (1.8)

INTERVENCIONES DE BIOMASA Y FLORA

Biomasa Crítica

Condición: Biomasa < `BAJA` **O** (flora_pred < flora_actual × 0,8 ∧ flora_actual < `FLORA_SPAWN_THRESHOLD`)
Intervención: `SPAWN_ENTITIES` (Flora)
Parámetros: Count = máx(0, `FLORA_SPAWN_THRESHOLD` − flora_actual), limitado a `MAXIMUM`

Ratio Flora/Herbívoro Bajo

Condición: $\frac{\text{flora}}{\text{herbívoro}} < 5{,}0 \wedge \text{herbívoros} > 0$
Intervención: `SPAWN_ENTITIES` (Flora)
Parámetros: Count = máx(anterior, herbívoros × 5,0 − flora_actual)

Flora en Declive Crítico

Condición: Flora < `CRÍTICA` ∧ tendencia_flora < `DECLIVE_MODERADO`
Intervención: `SPAWN_ENTITIES` (Flora)
Parámetros: Count = `MEDIUM`

BALANCE PREDADOR-PRESA (SIMBÓLICO)

Dominancia de Predadores

Condición: Balance = `PREDATOR_DOMINANT`
Intervenciones secuenciales:

1. `ADJUST_EVOLUTION` presas (si disponibles) - Factor: `ACCELERATE_EXTREME`
2. `ADJUST_EVOLUTION` depredadores (si disponibles) - Factor: `SLOW_SIGNIFICANT`
3. `SPAWN_ENTITIES` presas solo si población_presas ≤ `PREY_CRITICAL`

Parámetros spawn: Count = `calcular_spawn_count`(mín(`MEDIUM`, `MAXIMUM`), `HERBIVORE`)

Dominancia de Presas

Condición: Balance = `PREY_DOMINANT`
Intervenciones secuenciales:

1. `ADJUST_EVOLUTION` depredadores - Factor: `ACCELERATE_EXTREME`
2. `SPAWN_ENTITIES` depredadores si: población_depredadores $<$ `PREDATOR_CRITICAL` **O** (población_presas $>$ `PREY_ABUNDANT` $\wedge$ población_depredadores $<$ `SPAWN_LOW`)

Parámetros spawn: Count base = `LOW`, si tendencia_depredador $<$ 0: mín(`MEDIUM`, `MAXIMUM`), Final = `calcular_spawn_count`(base, `CARNIVORE`)

INTERVENCIONES HÍBRIDAS (NEURAL + SIMBÓLICO)

Spawn herbívoros por predicción

Condición: Ratio actual $>$ `PREDATOR_DOMINANT` $\wedge$ tendencia_presas $<$ `DECLIVE_MODERADO`
Intervención: `SPAWN_ENTITIES` (Herbívoros)
Parámetros: Count = `MEDIUM`

Spawn carnívoros por abundancia

Condición: Ratio actual $<$ `PREY_DOMINANT` $\wedge$ tendencia_presas $>$ `AUMENTO_SIG` $\wedge$ presas $>$ `ABUNDANTE`
Intervención: `SPAWN_ENTITIES` (Carnívoros)
Parámetros: Count = `calcular_spawn_count`(`MINIMAL`, CARNIVORE)

Ajuste por desviación predictiva

Condición: $\frac{|\text{ratio\_previsto} - \text{ratio\_actual}|}{\text{ratio\_actual}} > 0{,}25 \wedge$ ambos ratios > 0
Intervención: `ADJUST_EVOLUTION_CYCLE`
Lógica:

- Si ratio_pred $>$ actual: acelerar presas (`ACCELERATE_MODERATE`)
- Si ratio_pred $<$ actual: acelerar depredadores (`ACCELERATE_MODERATE`)

Flora crítica híbrida

Condición: Flora_actual $<$ `FLORA_CRÍTICA` $\wedge$ tendencia_flora $<$ `DECLIVE_MODERADO`
Intervención: `SPAWN_ENTITIES` (Flora)
Parámetros: Count = `MEDIUM`

Ajuste por estrés predictivo

Condición: avg_stress_pred $>$ avg_stress_actual $+$ `STRESS_THRESHOLD_SIG_INC` $\wedge$ species_stress disponible
Intervención: `ADJUST_EVOLUTION_CYCLE`
Parámetros: Especie = máx(species_stress.items()), Factor = `ACCELERATE_SIGNIFICANT`

ANÁLISIS DE GRAFOS Y ESTRUCTURA TRÓFICA

Protección especie Keystone

Condición: Keystone species detectadas $\wedge$ especie $\in$ `available_species`
Intervención: `ADJUST_EVOLUTION_CYCLE`
Parámetros: Factor = `ACCELERATE_SIGNIFICANT`, aplicado a cada especie keystone válida

---

Diversificación trófica

Condición: levels_count $< 3 \wedge$ niche_diversity $=$ `low` $\wedge$ depredadores disponibles
Intervención: `SPAWN_ENTITIES` (Carnívoros)
Parámetros: Count = `MINIMAL`
Objetivo: Incrementar complejidad de la red trófica (que a su vez, incrementa la conectividad)

ANÁLISIS Y AMPLIFICACIÓN DE TENDENCIAS

Amplificación por tendencias fuertes

Condición: |tendencia_presas| $>$ `MODERATE_INCREASE` **O** |tendencia_depredadores| $>$ `MODERATE_INCREASE`
Efecto: Aplicar amplification_factor $=$ `SIGNIFICANT` a intervenciones subsecuentes
Propósito: Sirve para reforzar las intervenciones cuando se detectan cambios grandes

---

Registro continuo de tendencias

Proceso: Evaluación continua del estado del bioima
Acción: Registra `integrated_result["trends"]` = {prey, predator, flora}
Uso: ES la base que luego me permitirá tomar decisiones futuras, o hacer análisis.

Comparativas resultados Reinforcement

Comparativa parcial con extractor por defecto de SB3

Todos los modelos presentes en el proyecto, los he entrenado en mi propio ordenador (con GPU NVIDIA GTX 1080ti). Muestro un entrenamiento con 250.000 steps:

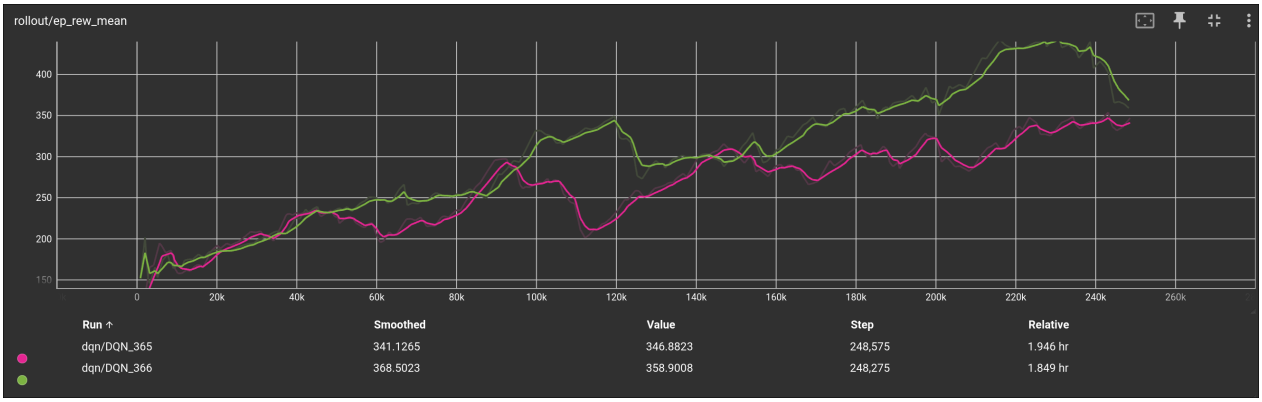


Figura 3: DQN_366, verde - extractor CNN customizado | DQN_365, rosa - extractor por defecto de SB3 (NatureCNN)

Como se puede observar, al comienzo no hay mucha diferencia, pero luego más adelante, hacia los 60k empieza a desmarcarse bastante el CNN customizado - intuyo que es porque a los embeddings les cuesta entrenar, y el motivo por cual se desmarca es ese. Nótese que SB3 por defecto, ya dispone una extractores de calidad y además altamente probados en muchos dominios (NatureCNN), con el mio simplemente me ha permitido refinar un poco más y hacer que generalice mejor. Tienen tendencias relativamente similares. Pero efectivamente, gracias a los embeddings y combinación de mapas, obtenemos mejores rewards.

Una última aventura de exploración

Una última aventura de exploración

Unos días antes de la entrega final, cedí ante la curiosidad y, por última vez antes de cerrar etapa, me dejé llevar por mi espíritu explorador. Pensé en poner mi `CNN` más parecida a como `SB3` implementa `NatureCNN`, pero manteniendo los cambios importantes que he hecho; combinación de mapas, embeddings etc (ver commit). Tenía puesto como **TO-DO** pendiente el incorporar una convolucional más, así como jugar con pesos etc, así que era algo que tenía en mente probar. Gorka Azkune, en la asignatura de AARN nos repitió múltiples veces: *si queréis probar a diseñar vosotros la arquitectura, probad, pero al final lo mejor es utilizar las que han sido diseñadas y testadas por otros*. Y efectivamente, tenía razón. Basándome en la implementación de `NatureCNN`, la cual a su vez se basa en el paper de DQN original (*Mnih, Kavukcuoglu, Silver, Rusu et al., 2015*[54]) decidí cambiar mi extractor de características, a última hora.

Básicamente ahora mi capa convolucional, **no tiene ni batchnorm ni maxpool** y he agregado una capa extra más convolucional (paso de 2 a 3). Con ello, recalculo por capa los tamaños de salidas, utilizando la fórmula que ya vimos en clase de AARN (función anidada `conv_out` en el commit). También cambio los tamaños de kernels de 3 - 2 (`maxpool`) - 3, a 8 - 4 - 3. El problema lo he tenido con los paddings, ya que `NatureCNN` trabaja con mapas de 84x84 y tener padding 0 le viene bien, no introduce relleno/ruído. En mi caso, utilizando FOVs relativamente pequeños como 17x17, el tamaño se rompe. Así que he tenido que jugar con los paddings y poner 2 - 1 - 1; los mínimos.

Tambien ahora utilizo inicialización **ortogonal**, como ellos. Y, como he comentado, mantengo el resto del extractor igual, con embeddings, combinaciones de mapas...etc. Veamos el resultado (naranja).

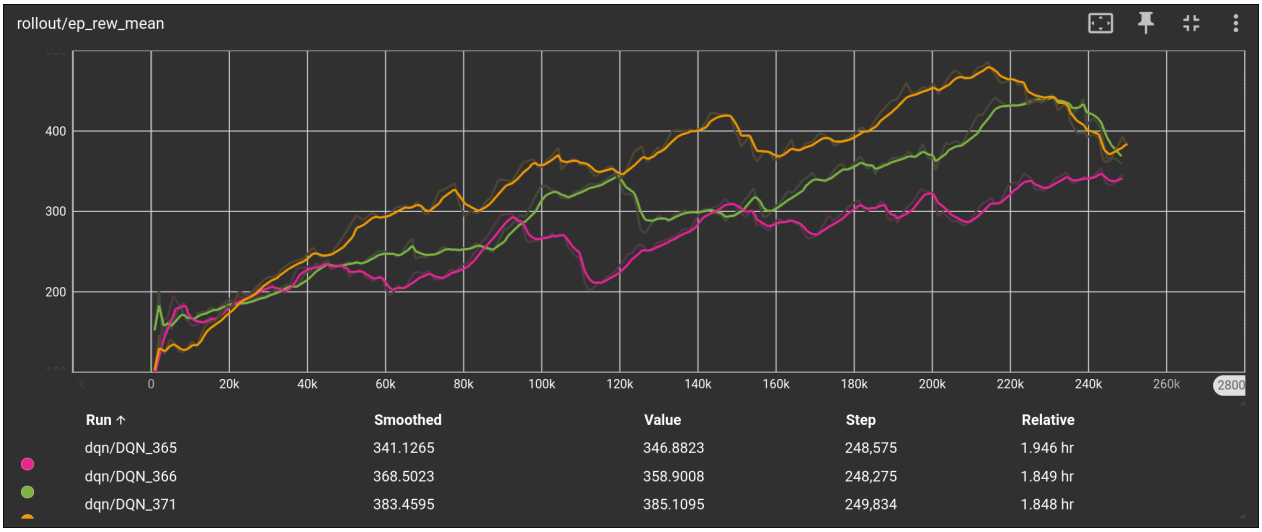
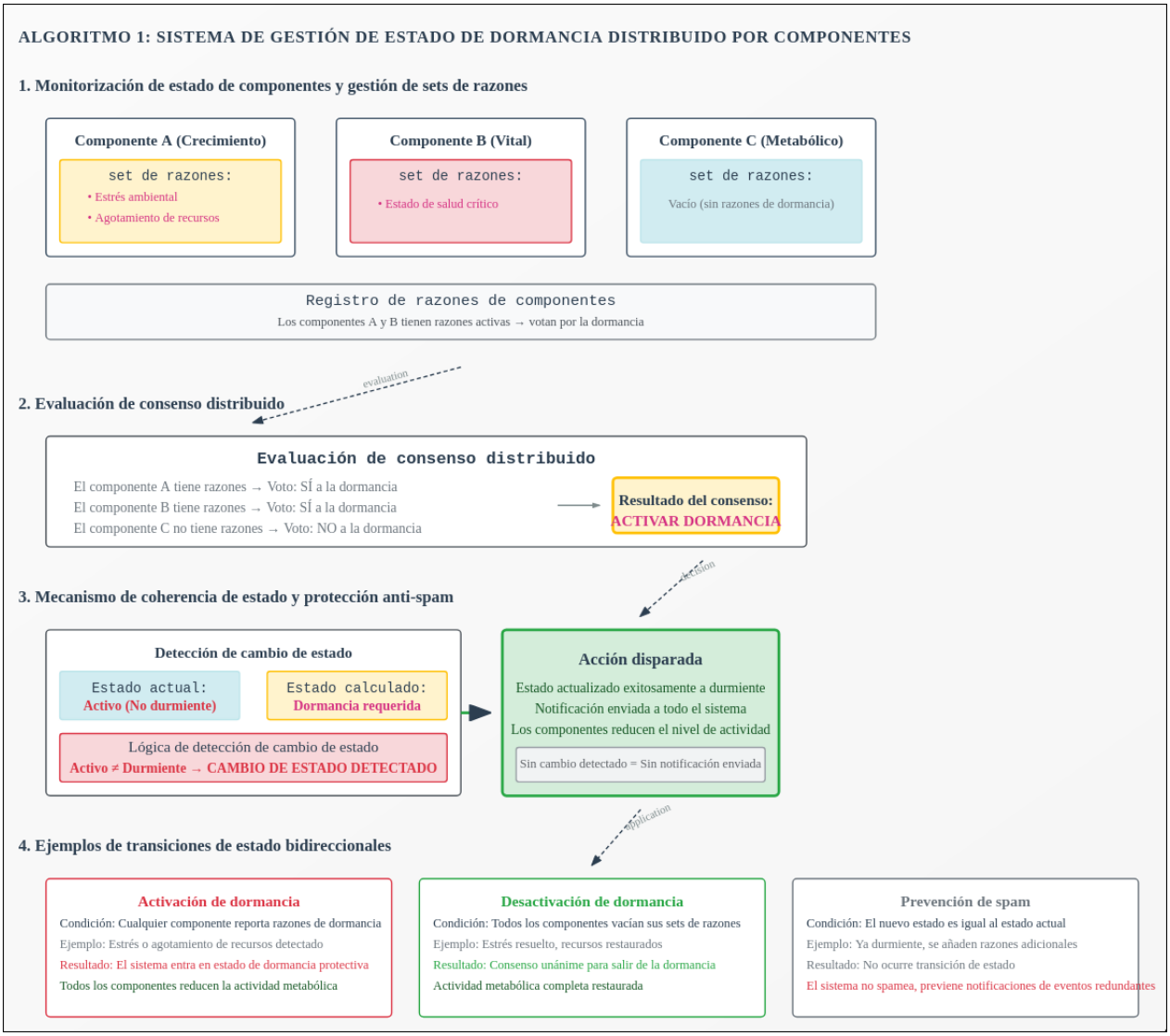


Figura 4: DQN_371, naranja – extractor basado en `NatureCNN` + mis aportaciones | DQN_366, verde – extractor CNN customizado | DQN_365, rosa – extractor por defecto de SB3 (`NatureCNN`)

Maravilloso, conseguimos un reward medio bastante más alto por lo general, otro salto en rendimiento.

Material gráfico complementario

Algoritmo de dormancia



Modelo de forrajeo

Ciclo de entrenamiento fauna con Reinforcement

Figura 5: Muestra de ciclo del training de fauna con reinforcement, múltiples biomas, targets etc.

Figura 6: Continuación. Muestra de ciclo del training de fauna con reinforcement, múltiples biomas, targets etc.

Descripciones

Descripciones de videos de entrenamiento de fauna

Por si hay algún problema con las descripciones en los videos de Youtube, *vuelco* directamente aquí las descripciones de cada uno, copio directamente:

(a) Génesis de Inteligencia: activación de comportamientos por cercanía

En este video muestro diferentes simulaciones extraídas del proceso de entrenamiento de fauna mediante Reinforcement Learning. La idea aquí es mostrar cómo se activan comportamientos en las cercanías del agente (target escogido para esta simulación). De ésta forma guío al agente mediante la función de rewards, para que aprenda estas dinámicas de presa/depredador dependiendo de su rol, y sea capaz de interiorizarlas.

(b) Cervatillo maestro del escape

En este video, ya en una simulación normal (modelos en modo inferencia), vemos las hazañas de un cervatillo, el cual está comiendo vegetación y explorando, pero es acechado varias veces, pero de una forma u otra consigue escapar. Además, no se olvida de beber agua cuando es necesario.

La simulación terminó antes de poder ver si es capaz de escapar también de esa última encerrona al final.

Para éste bioma y escosistema:

- Círculos negros - depredadores (panteras, carnívoros)
- Círculos marrones clarito - presas (cervatillos, herbívoros)
- El resto es flora.
- Simulación con escenario simplificado para ilustrar los comportamientos.

(c) Estrategias de acorralamiento

Aquí podemos ver a dos panteras acorralar a 3 cervatillos, aunque uno termina escapando, a los otros dos les es imposible y caen en la letal encerrona - sirven de alimento para los depredadores.

Para éste bioma y escosistema:

- Círculos negros - depredadores (panteras, carnívoros)
- Círculos marrones clarito - presas (cervatillos, herbívoros)
- El resto es flora.
- Simulación con escenario simplificado para ilustrar los comportamientos.

(d) Huída por falta de energía

Cuando una entidad se queda sin reservas de energía, ya no puede moverse. Pues en este video vemos cómo un cervatillo que está siendo perseguido por una pantera que, intentando recuperar energía le está intentando cazar, consigue escapar gracias a ésta se queda sin reservas durante la caza.

Para éste bioma y escosistema:

- Círculos negros - depredadores (panteras, carnívoros)
- Círculos marrones clarito - presas (cervatillos, herbívoros)
- El resto es flora.
- Simulación con escenario simplificado para ilustrar los comportamientos.

(e) Dispersión de manada + target lock

Este escenario es un poco diferente a los previos, pero muy, muy interesante. Nada más comenzar, la pantera seleccionada tiene dos opciones: 1) ir a por cervatillo que está abajo a su derecha, o ir hacia la masa, la manada de cervatillos a la izquierda. Toma la opción que más probable es le de alimento, que es ir hacia la manada. Ahora, algo muy interesante es ver dispersarse a estos y huir de la pantera para no ser cazados. Ésta, habiendo perdido la oportunidad, sigue explorando pero al poco vé otra víctima, a la cual persigue de muy notable hasta que, termina arrinconándola y

alimentandose de ella.

Para éste bioma y escosistema:

- Círculos negros - depredadores (panteras, carnívoros)
- Círculos marrones clarito - presas (cervatillos, herbívoros)
- El resto es flora.
- Simulación con escenario simplificado para ilustrar los comportamientos.

(f) Consecuencias 0 depredadores, desequilibrio ecológico

Para este escenario, no hay depredadores - la idea es poder observar qué ocurre si se rompe el ciclo que mantiene los ecosistemas, a un nivel básico.

Como podemos observar, al no haber depredadores, los herbívoros (cervatillos) arrasan con toda la flora, al no haber flora suficiente para mitigar el CO2 que los herviboros emiten con su respiración, éste sube sobremanera, lo cual tiene efecto a largo plazo en el clima y contribuye al efecto invernadero. Además, al arrasar con la flora, el ciclo de reproducción de la misma no es suficiente para poder alimentar a los herbívoros, y la población cae en picado al no haber sustento. Esto nos demuestra que un desequilibrio en la cadena trófica, lleva a una cadena de efectos en cascada.

Al final, cuando no quedan casi herviboros, la flora comienza a florecer más continuamente y se mejoran los niveles de CO2. Pero, tenemos un ecosistema insostenible.

Para éste bioma y escosistema:

- Sin depredadores
- Círculos marrones clarito - presas (cervatillos, herbívoros)
- El resto es flora.
- Simulación con escenario simplificado para ilustrar los comportamientos.

(g) Depredador sobreviviendo

En éste último video, tenemos una pantera en relativa armonia; pasa su tiempo sobreviviendo de manera bastante exitosa; tiene agua cuando la necesita cerca y hace uso de ella para hidratarse, se permite explorar y cazar herviboros que merodean por la zona. Incluso para recargar energias se introduce en un bosque de algo más al norte.

Para éste bioma y escosistema:

- Círculos negros - depredadores (panteras, carnívoros)
- Círculos marrones clarito - presas (cervatillos, herbívoros)
- El resto es flora.
- Simulación con escenario simplificado para ilustrar los comportamientos.

Bibliografía

[1] Thomas S. Ray. «An approach to the synthesis of life». En: *Artificial Life II* (1991), págs. 371-408.

[2] Charles Ofria y Claus O. Wilke. «Avida: A software platform for research in computational evolutionary biology». En: *Artificial Life* 10.2 (2004), págs. 191-229.

[3] ALIEN Project. *ALIEN - Artificial Life Environment*. https://alien-project.org/. Accessed: 2025. 2024.

[4] Jacques Gignoux, Ian D. Davies y Shayne R. Flint. «3Worlds, a simulation platform for ecosystem modelling». En: *Ecological Modelling* 473 (2022), pág. 110121.

[5] B. Layman. *Artificial-Life-Simulator: An ecologically inspired multi-agent system. Agents are designed with neural network based decision making, and complex resource requirements*. https://github.com/BLayman/Artificial-Life-Simulator. Accessed: 2025. 2019.

[6] Nakamasa Inoue, Eisuke Yamagata e Hirokatsu Kataoka. *Initialization Using Perlin Noise for Training Networks with a Limited Amount of Data*. 2021. arXiv: 2101.07406 [cs.CV].

[7] Ken Perlin. «An Image Synthesizer». En: *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '85. New York, NY, USA: ACM, 1985, págs. 287-296. DOI: 10.1145/325165.325247.

[8] Ken Perlin. «Improving Noise». En: *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '02. San Antonio, Texas: ACM, 2002, págs. 681-682. ISBN: 1-58113-521-1. DOI: 10.1145/566570.566636.

[9] Stéfan van der Walt, S. Chris Colbert y Gaël Varoquaux. «The NumPy Array: A Structure for Efficient Numerical Computation». En: *IEEE Computing in Science & Engineering* 13.2 (mar. de 2011), págs. 22-30. DOI: 10.1109/MCSE.2011.37.

[10] Pauli Virtanen et al. «SciPy 1.0: fundamental algorithms for scientific computing in Python». En: *Nature Methods* 17.3 (mar. de 2020), págs. 261-272. DOI: 10.1038/s41592-019-0686-2.

[11] Owen K. Atkin y Mark G. Tjoelker. «Thermal acclimation and the dynamic response of plant respiration to temperature». En: *Trends in Plant Science* 8.7 (2003), págs. 343-351.

[12] Richard G. Allen et al. *Crop Evapotranspiration – Guidelines for Computing Crop Water Requirements*. FAO Irrigation and Drainage Paper 56. Food y Agriculture Organization of the United Nations. Rome, Italy, 1998.

[13] Intergovernmental Panel on Climate Change (IPCC). *Climate Change 2013: The Physical Science Basis. Contribution of Working Group I to the Fifth Assessment Report*. Inf. téc. Available online: https://www.ipcc.ch/report/ar5/wg1/. Cambridge, UK y New York, NY, USA: IPCC, 2013.

[14] NASA. *The Global Carbon Cycle*. https://earthobservatory.nasa.gov/features/CarbonCycle. Accessed: 2025-06-07. 2011.

[15] G. D. Farquhar, S. von Caemmerer y J. A. Berry. «A biochemical model of photosynthetic CO_2 assimilation in leaves of C_3 species». En: *Planta* 149.1 (1980), págs. 78-90.

[16] J. Flexas et al. «Diffusive and metabolic limitations to photosynthesis under drought and salinity in C_3 plants». En: *Plant Biology* 6.3 (2004), págs. 269-279.

[17] F. S. Chapin III, P. A. Matson y P. M. Vitousek. *Principles of Terrestrial Ecosystem Ecology*. 2nd. Springer, 2011.

[18] Benjamin Gompertz. «On the nature of the function expressive of the law of human mortality, and on a new mode of determining the value of life contingencies». En: *Philosophical Transactions of the Royal Society of London* 115 (1825), págs. 513-585.

[19] A. Golubev. «How could the Gompertz–Makeham law evolve». En: *Journal of Theoretical Biology* 258.1 (2009), págs. 1-17. DOI: 10.1016/j.jtbi.2009.01.009.

[20] James W. Vaupel et al. «Biodemographic trajectories of longevity». En: *Science* 280.5365 (1998), págs. 855-860. DOI: 10.1126/science.280.5365.855.

[21] Edward J. Calabrese y Robyn Blain. «The occurrence of hormetic dose responses in the toxicological literature, the hormesis database: an overview». En: *Toxicology and Applied Pharmacology* 202.3 (2005), págs. 289-301. DOI: 10.1016/j.taap.2004.06.023.

[22] Hans Lambers, F. Stuart Chapin y Thijs L. Pons. *Plant Physiological Ecology*. 2.ª ed. New York: Springer, 2008. ISBN: 978-0-387-78341-3.

[23] Arnold J. Bloom, F. Stuart Chapin y Harold A. Mooney. «Resource limitation in plants: an economic analogy». En: *Annual Review of Ecology and Systematics* 16 (1985), págs. 363-392. DOI: 10.1146/annurev.es.16.110185.002051.

[24] Marcel Van Oijen, Ad H. C. M. Schapendonk y Mats Höglind. «On the relative magnitudes of photosynthesis, respiration, growth and carbon storage in vegetation». En: *Annals of Botany* 105.5 (2010), págs. 793-797. DOI: 10.1093/aob/mcq039.

[25] J. S. Amthor. «The McCree–de Wit–Penning de Vries–Thornley Respiration Paradigms: 30 Years Later». En: *Annals of Botany* 86.1 (2000), págs. 1-20. DOI: 10.1006/anbo.2000.1175.

[26] Ü. Niinemets. «Stomatal conductance alone does not explain the decline in foliar photosynthetic rates with increasing tree age and size in Picea abies and Pinus sylvestris». En: *Tree Physiology* 22.8 (2002), págs. 515-535. DOI: 10.1093/treephys/22.8.515.

[27] Eugene D. Schneider y James J. Kay. «Life as a manifestation of the second law of thermodynamics». En: *Mathematical and Computer Modelling* 19.6-8 (1994), págs. 25-48. DOI: 10.1016/0895-7177(94)90188-0.

[28] Sharon N. Greenwood, Regina G. Belz y Brian P. Weiser. «A Conserved Mechanism for Hormesis in Molecular Systems». En: *DoseResponse* 20.3 (2022), pág. 15593258221109335. DOI: 10.1177/15593258221109335.

[29] Nancy C. Johnson. «Resource stoichiometry elucidates the structure and function of arbuscular mycorrhizas across scales». En: *New Phytologist* 185.3 (2010), págs. 631-647. DOI: 10.1111/j.1469-8137.2009.03110.x.

[30] John P. Bryant, F. Stuart III Chapin y David R. Klein. «Carbon/nutrient balance of boreal plants in relation to vertebrate herbivory». En: *Oikos* 40.3 (1983), págs. 357-368. DOI: 10.2307/3544308.

[31] Nico M. van Straalen y T. F. M. Roelofs. *An Introduction to Ecological Genomics*. Oxford, UK: Oxford University Press, 2006. ISBN: 978-0198566717.

[32] Bruce S. McEwen. «Allostasis and allostatic load: implications for neuropsychopharmacology». En: *Neuropsychopharmacology* 22.2 (2000), págs. 108-124. DOI: 10.1016/S0893-133X(99)00129-3.

[33] Mikko Känkänen y Juhani Pirhonen. «The effect of intermittent feeding on feed intake and compensatory growth of whitefish *Coregonus lavaretus* L.» En: *Aquaculture* 288.1–2 (2009), págs. 92-97. DOI: 10.1016/j.aquaculture.2008.11.029.

[34] Mohamed Ali, Almudena Nicieza y Richard J. Wootton. «Compensatory growth in fishes: A response to growth depression». En: *Fish and Fisheries* 4.2 (2003), págs. 147-190. DOI: 10.1046/j.1467-2979.2003.00120.x.

[35] Garrick T. Skalski et al. «Variable intake, compensatory growth, and increased growth efficiency in fish: Models and mechanisms». En: *Ecology* 86.6 (2005), págs. 1452-1462. DOI: 10.1890/04-0896.

[36] Walter Larcher. *Physiological Plant Ecology: Ecophysiology and Stress Physiology of Functional Groups*. 3.ª ed. Berlin: Springer Science & Business Media, 2003.

[37] Zejun Zhang et al. «Faster or Slower? Performance Mystery of Python Idioms Unveiled with Empirical Evidence». En: *Proceedings of the 45th IEEE/ACM International Conference on Software Engineering (ICSE)*. IEEE/ACM, 2023.

[38] R. H. Whittaker. *Communities and Ecosystems*. 2nd. New York, NY: Macmillan, 1975.

[39] R. L. Lindeman. «The trophic-dynamic aspect of ecology». En: *Ecology* 23.4 (1942), págs. 399-417.

[40] Alex Rogers y Adam Prügel-Bennett. «Evolving Populations with Overlapping Generations». En: *Theoretical Population Biology* 57.2 (mar. de 2000), págs. 121-129. DOI: 10.1006/tpbi.1999.1446.

[41] Agoston E. Eiben y James E. Smith. *Introduction to Evolutionary Computing*. 2.ª ed. Natural Computing Series. Berlin, Heidelberg: Springer, 2015. ISBN: 978-3-662-44874-8. DOI: 10.1007/978-3-662-44874-8.

[42] Agoston E. Eiben, Robert Hinterding y Zbigniew Michalewicz. «Parameter Control in Evolutionary Algorithms». En: *IEEE Transactions on Evolutionary Computation* 3.2 (jul. de 1999), págs. 124-141. DOI: 10.1109/4235.771166.

[43] Alex Rogers y Adam Prügel-Bennett. «Evolving Populations with Overlapping Generations». En: *Theoretical Population Biology* 57.2 (2000), págs. 121-129. DOI: 10.1006/tpbi.1999.1446.

[44] John Schulman et al. «Proximal Policy Optimization Algorithms». En: *arXiv preprint arXiv:1707.06347* (2017). URL: https://arxiv.org/abs/1707.06347.

[45] Stable Baselines3 Contributors. *PPO — Stable Baselines3 Documentation*. 2025. URL: https://stable-baselines3.readthedocs.io/en/master/modules/ppo.html (visitado 21-05-2025).

[46] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu et al. «Human-level control through deep reinforcement learning». En: *Nature* 518.7540 (2015), págs. 529-533. DOI: 10.1038/nature14236.

[47] Peter Henderson et al. «Deep Reinforcement Learning that Matters». En: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*. AAAI Press, 2018, págs. 3207-3214. DOI: 10.1609/aaai.v32i1.11694.

[48] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves et al. «Playing Atari with Deep Reinforcement Learning». En: *arXiv preprint arXiv:1312.5602* (2013).

[49] Stephane Ross, Geoffrey J. Gordon y J. Andrew Bagnell. «A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning». En: *Journal of Machine Learning Research* 15 (2011), págs. 627-635.

[50] Binyu Wang et al. «Mobile Robot Path Planning in Dynamic Environments through Globally Guided Reinforcement Learning». En: *arXiv preprint arXiv:2005.05420v2* (2020). Submitted 11 Sep 2020. arXiv: 2005.05420v2 [cs.RO].

[51] E. E. Moore, M. A. Johnson y J. R. Smith. *Thirst sensations and fluid ingestion following graded hypohydration in human volunteers*. 1987.

[52] J. A. Roberts, K. Lee y R. S. Patel. *Objective assessment of thirst and hydration state in rhesus monkeys*. 2005.

[53] Michaël Mathieu et al. *AlphaStar Unplugged: Large-Scale Offline Reinforcement Learning*. 2023. arXiv: 2308.03526 [cs.LG]. URL: https://arxiv.org/abs/2308.03526.

[54] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu et al. «Human-level control through deep reinforcement learning». En: *Nature* 518.7540 (2015), págs. 529-533. DOI: 10.1038/nature14236.

[55] Chen Liang et al. «Neural Symbolic Machines: Learning Semantic Parsers on Freebase with Weak Supervision». En: *arXiv preprint arXiv:1611.00020* (2017). version 4, 23 April 2017. arXiv: 1611.00020 [cs.CL].

[56] A. J. Lotka. «Analytical Note on Certain Rhythmic Relations in Organic Systems». En: *Proceedings of the National Academy of Sciences of the United States of America* 6.7 (1920), págs. 410-415.

[57] V. Volterra. «Fluctuations in the Abundance of a Species Considered Mathematically». En: *Nature* 118.2972 (1926), págs. 558-560.

[58] Joel E. Cohen, François Briand y Charles M. Newman. *Community Food Webs: Data and Theory*. Berlin: Springer-Verlag, 1990.

[59] Ronald Aylmer Fisher. *Statistical Methods for Research Workers*. 1st. Edinburgh: Oliver y Boyd, 1925.

[60] Ronald Aylmer Fisher. *The Design of Experiments*. Edinburgh: Oliver y Boyd, 1935.

[61] Ashish Vaswani et al. «Attention Is All You Need». En: *Advances in Neural Information Processing Systems*. Vol. 30. 2017, págs. 5998-6008. URL: https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

[62] Shaojie Bai, J. Zico Kolter y Vladlen Koltun. «An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling». En: *arXiv:1803.01271* (2018).

[63] Paul Suganthan et al. «Adapting Decoder-Based Language Models for Diverse Encoder Downstream Tasks». En: *arXiv preprint arXiv:2503.12345* (mar. de 2025). URL: https://arxiv.org/abs/2503.12345.

[64] Mohammad Irani Azad et al. «Sequence-to-Sequence Model with Transformer-based Attention Mechanism and Temporal Pooling for Non-Intrusive Load Monitoring». En: *arXiv preprint arXiv:2306.05012* (2023). URL: https://arxiv.org/abs/2306.05012.

[65] Dongyue Guo et al. «A Non-autoregressive Multi-Horizon Flight Trajectory Prediction Framework with Gray Code Representation». En: *AAAI Conference on Artificial Intelligence (AAAI)*. 2023. URL: https://arxiv.org/abs/2305.01658.

[66] Ole Henrik Nordvik. «A Horizon Metadata Transformer for Multi-Horizon Forecasting». Tesis de mtría. Norwegian University of Science y Technology, 2024. URL: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/3026841.

[67] Rob J. Hyndman y George Athanasopoulos. *Forecasting: Principles and Practice*. 2nd. OTexts, 2018. URL: https://otexts.com/fpp2/.

[68] Boje Deforce, Bart Baesens y Estefanía Serral Asensio. «Time-Series Foundation Models for Forecasting Soil Moisture Levels in Smart Agriculture». En: *Proceedings of the KDD '24 Fragile Earth Workshop*. arXiv:2405.18913. Barcelona, Spain, ago. de 2024, pág. 7.

[69] Alex Kendall, Yarin Gal y Roberto Cipolla. «Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics». En: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, págs. 7482-7491. DOI: 10.1109/CVPR.2018.00781.

[70] Pratyush Maini et al. «Why and when should you pool? Analyzing Pooling in Recurrent Architectures». En: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics, 2020, págs. 4568-4586. DOI: 10.18653/v1/2020.findings-emnlp.410.

[71] Peng Zhou et al. «Text Classification Improved by Integrating Bidirectional LSTM with Two-dimensional Max Pooling». En: *CoRR* abs/1611.06639 (2016). arXiv: 1611.06639. URL: http://arxiv.org/abs/1611.06639.

[72] Bryan Lim et al. «Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting». En: *International Journal of Forecasting* 37.4 (2021), págs. 1748-1764. DOI: 10.1016/j.ijforecast.2021.03.012.